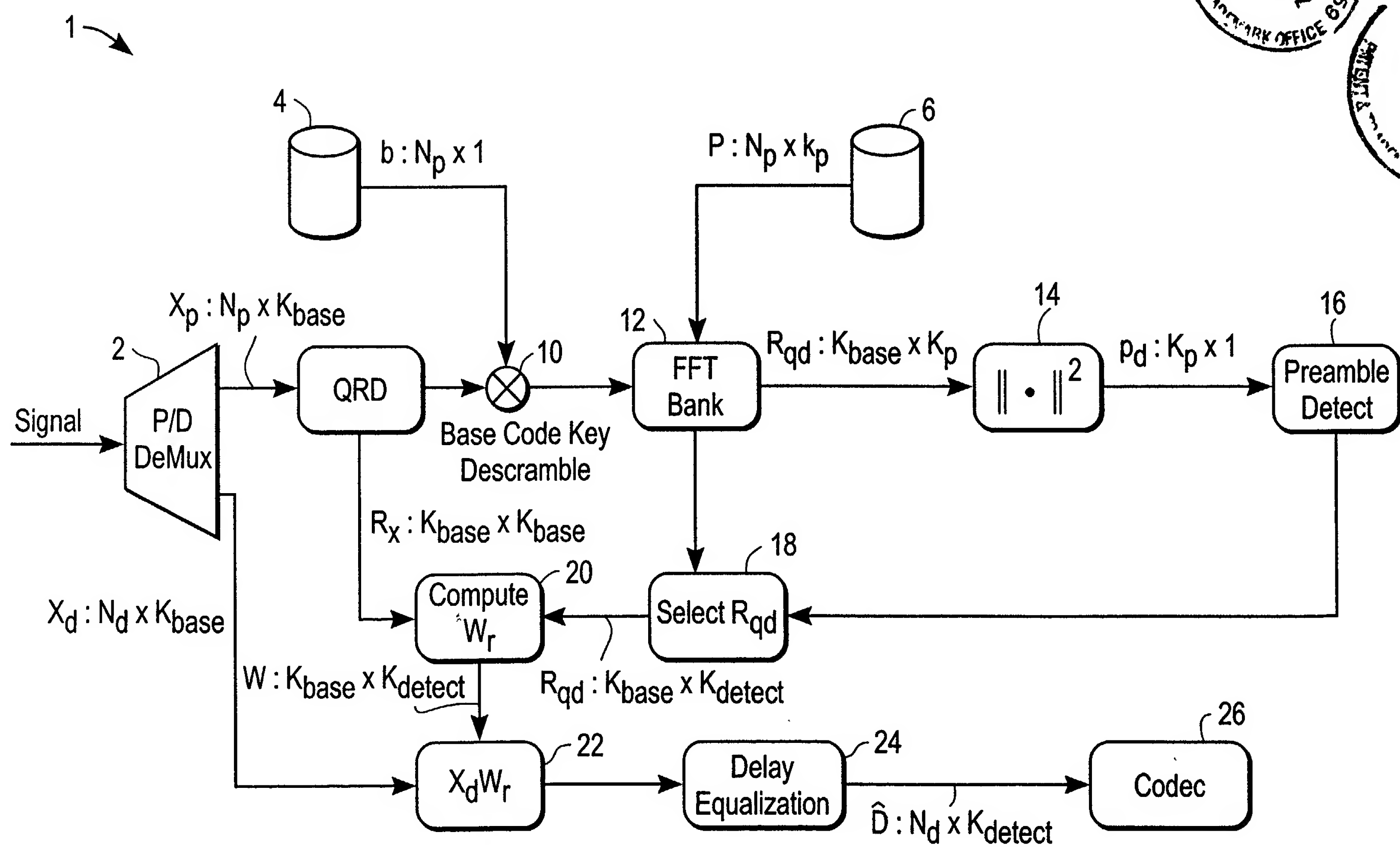

1
Signal
2
P/D DeMux
Xp : Np x Kbase
Xd : Nd x Kbase
4
b : Np x 1
QRD
10
Base Code Key Descramble
Rx : Kbase x Kbase
6
P : Np x kp
12
FFT Bank
Rqd : Kbase x Kp
14
‖ • ‖2
pd : Kp x 1
16
Preamble Detect
18
Select Rqd
Rqd : Kbase x Kdetect
20
Compute Wr
W : Kbase x Kdetect
22
XdWr
24
Delay Equalization
D̂ : Nd x Kdetect
26
Codec

FIG. 1

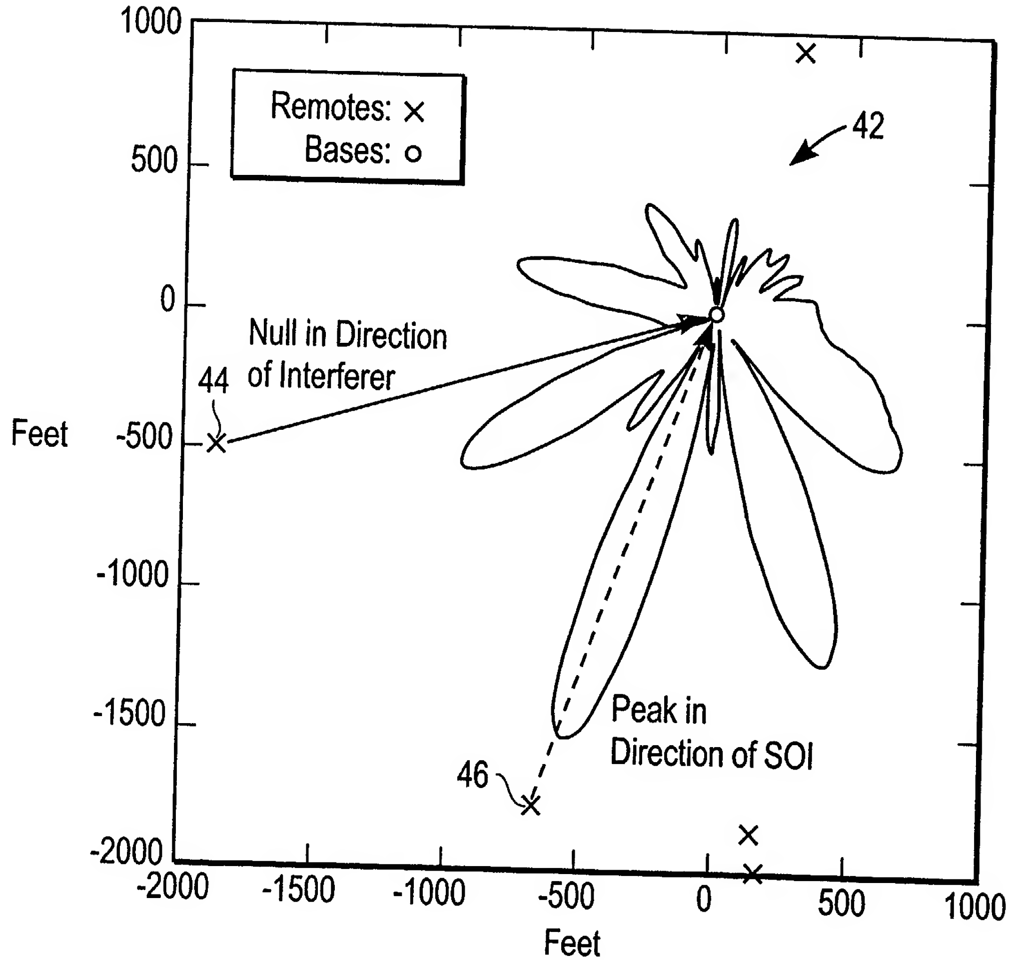

Remotes: ×
Bases: o
42
Null in Direction
of Interferer
44
Peak in
Direction of SOI
46
1000
500
0
-500
-1000
-1500
-2000
Feet
-2000
-1500
-1000
-500
0
500
1000
Feet

FIG. 2

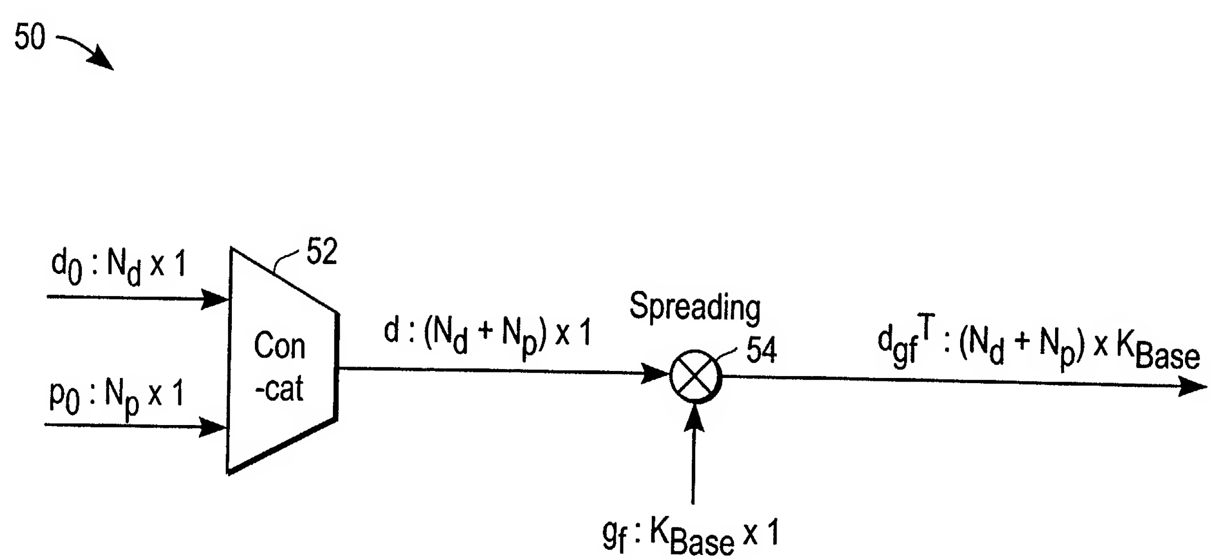

50
52
Con
-cat
$d_0 : N_d \times 1$
$p_0 : N_p \times 1$
$d : (N_d + N_p) \times 1$
Spreading
54
$g_f : K_{Base} \times 1$
$d_{gf}{}^T : (N_d + N_p) \times K_{Base}$

FIG. 3

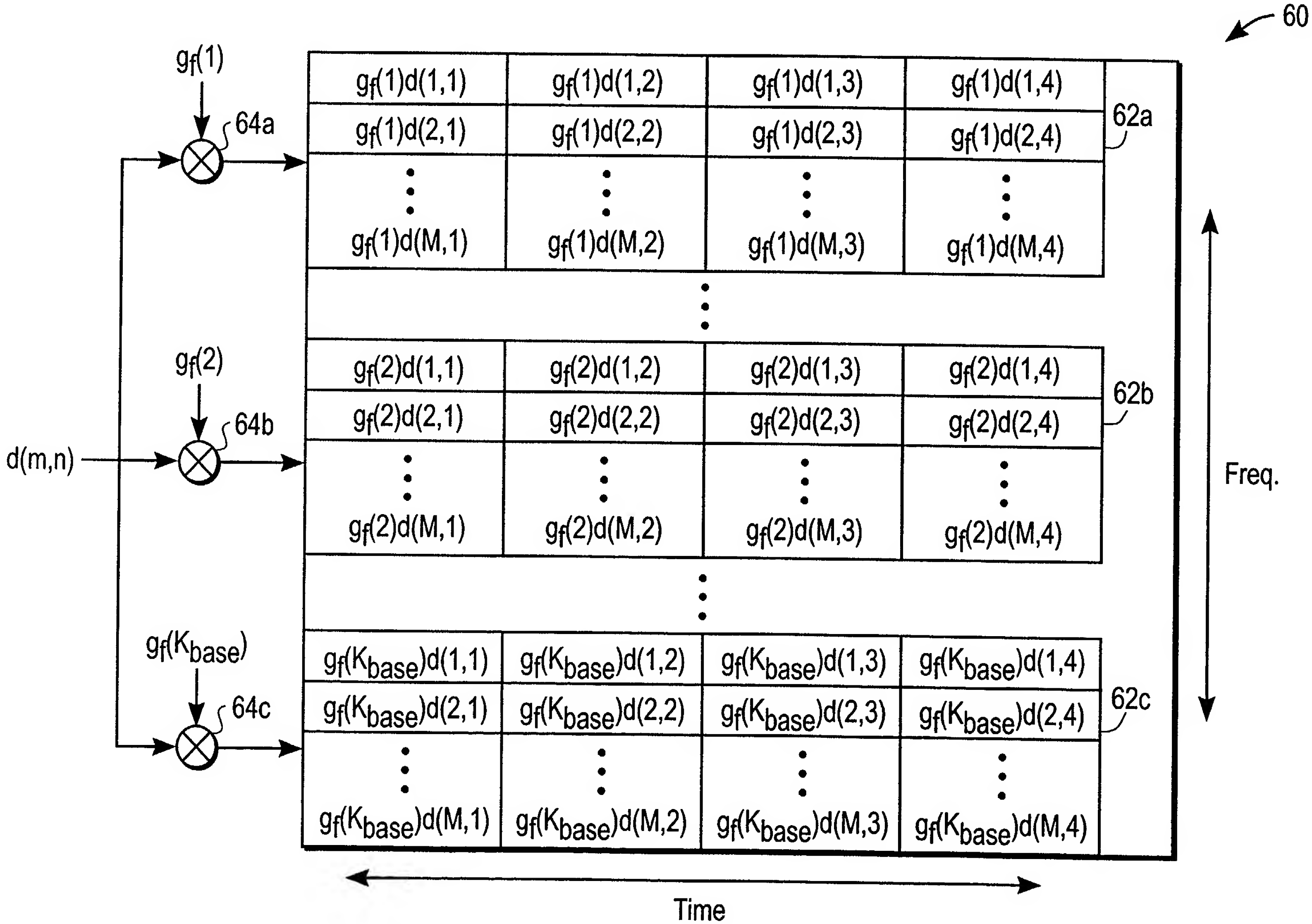

60
$g_f(1)$
$g_f(2)$
$g_f(K_{base})$
64a
64b
64c
d(m,n)
$g_f(1)d(1,1)$
$g_f(1)d(1,2)$
$g_f(1)d(1,3)$
$g_f(1)d(1,4)$
$g_f(1)d(2,1)$
$g_f(1)d(2,2)$
$g_f(1)d(2,3)$
$g_f(1)d(2,4)$
$g_f(1)d(M,1)$
$g_f(1)d(M,2)$
$g_f(1)d(M,3)$
$g_f(1)d(M,4)$
62a
$g_f(2)d(1,1)$
$g_f(2)d(1,2)$
$g_f(2)d(1,3)$
$g_f(2)d(1,4)$
$g_f(2)d(2,1)$
$g_f(2)d(2,2)$
$g_f(2)d(2,3)$
$g_f(2)d(2,4)$
$g_f(2)d(M,1)$
$g_f(2)d(M,2)$
$g_f(2)d(M,3)$
$g_f(2)d(M,4)$
62b
$g_f(K_{base})d(1,1)$
$g_f(K_{base})d(1,2)$
$g_f(K_{base})d(1,3)$
$g_f(K_{base})d(1,4)$
$g_f(K_{base})d(2,1)$
$g_f(K_{base})d(2,2)$
$g_f(K_{base})d(2,3)$
$g_f(K_{base})d(2,4)$
$g_f(K_{base})d(M,1)$
$g_f(K_{base})d(M,2)$
$g_f(K_{base})d(M,3)$
$g_f(K_{base})d(M,4)$
62c
Freq.
Time

FIG. 4

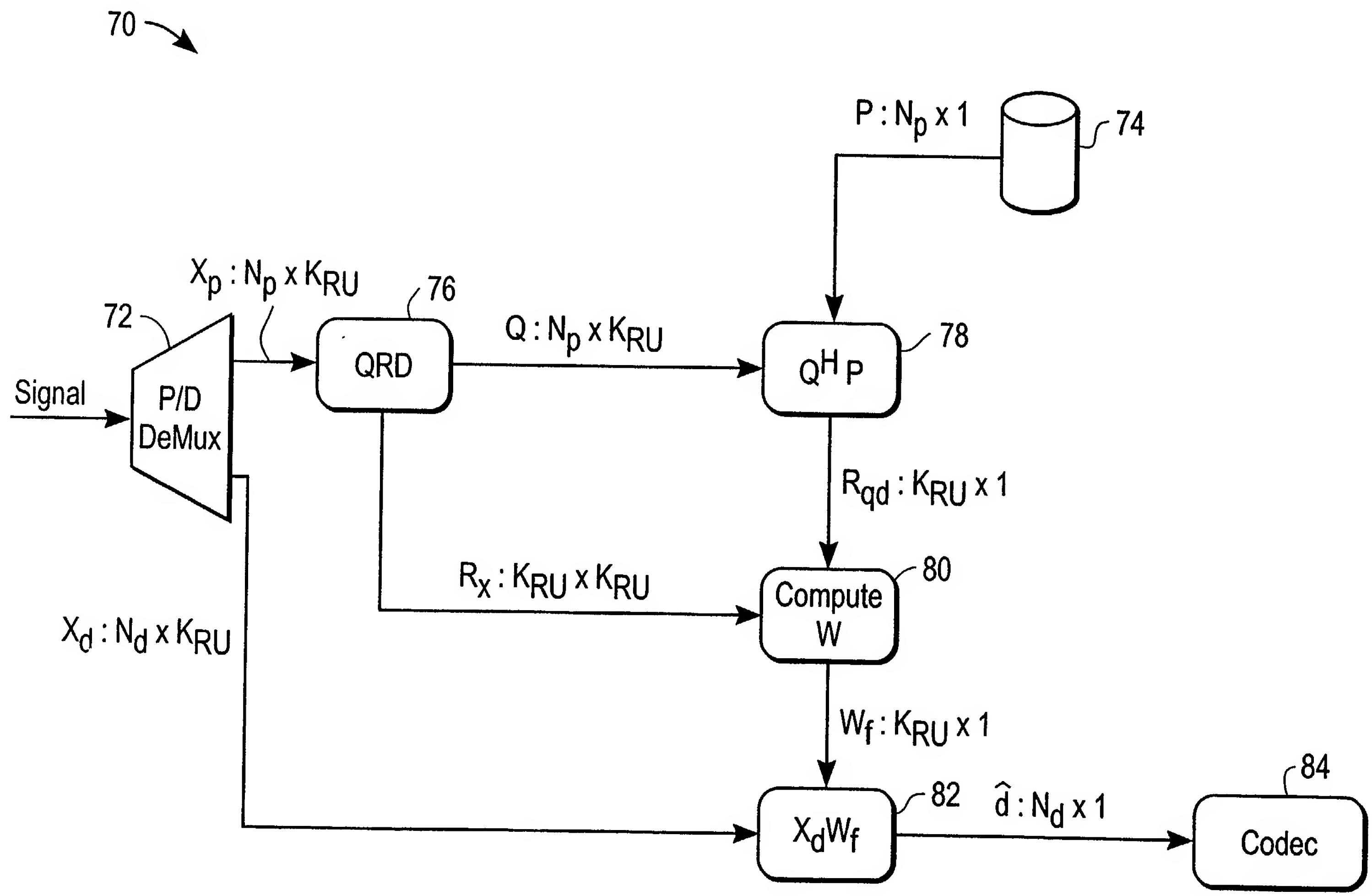

70
Signal
72
P/D
DeMux
$X_p : N_p \times K_{RU}$
76
QRD
$Q : N_p \times K_{RU}$
$P : N_p \times 1$
74
78
$Q^H P$
$R_{qd} : K_{RU} \times 1$
$R_x : K_{RU} \times K_{RU}$
80
Compute
W
$X_d : N_d \times K_{RU}$
$W_f : K_{RU} \times 1$
82
$X_d W_f$
$\hat{d} : N_d \times 1$
84
Codec

FIG. 5

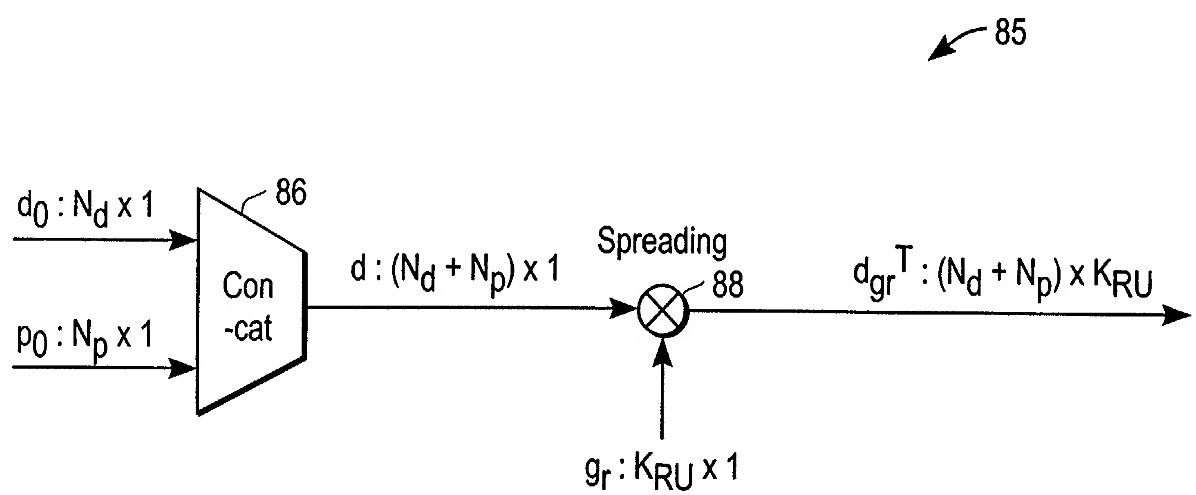

85
$d_0 : N_d \times 1$
$p_0 : N_p \times 1$
86
Con
-cat
$d : (N_d + N_p) \times 1$
Spreading
88
$g_r : K_{RU} \times 1$
$d_{gr}^T : (N_d + N_p) \times K_{RU}$

FIG. 6

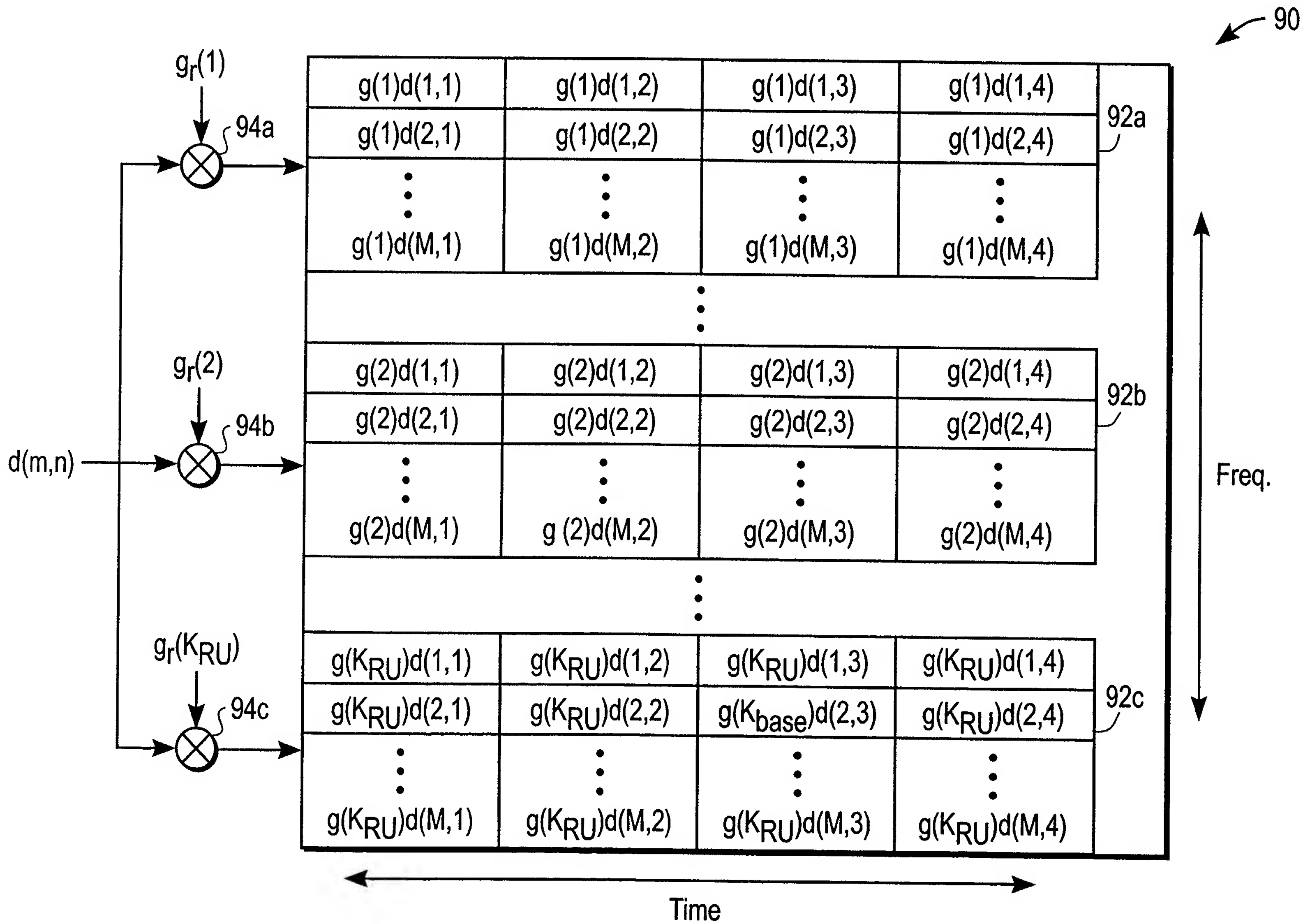

90
$g_r(1)$
94a
$g_r(2)$
94b
$g_r(K_{RU})$
94c
d(m,n)
g(1)d(1,1)
g(1)d(1,2)
g(1)d(1,3)
g(1)d(1,4)
g(1)d(2,1)
g(1)d(2,2)
g(1)d(2,3)
g(1)d(2,4)
g(1)d(M,1)
g(1)d(M,2)
g(1)d(M,3)
g(1)d(M,4)
92a
g(2)d(1,1)
g(2)d(1,2)
g(2)d(1,3)
g(2)d(1,4)
g(2)d(2,1)
g(2)d(2,2)
g(2)d(2,3)
g(2)d(2,4)
g(2)d(M,1)
g (2)d(M,2)
g(2)d(M,3)
g(2)d(M,4)
92b
$g(K_{RU})d(1,1)$
$g(K_{RU})d(1,2)$
$g(K_{RU})d(1,3)$
$g(K_{RU})d(1,4)$
$g(K_{RU})d(2,1)$
$g(K_{RU})d(2,2)$
$g(K_{base})d(2,3)$
$g(K_{RU})d(2,4)$
$g(K_{RU})d(M,1)$
$g(K_{RU})d(M,2)$
$g(K_{RU})d(M,3)$
$g(K_{RU})d(M,4)$
92c
Freq.
Time

FIG. 7

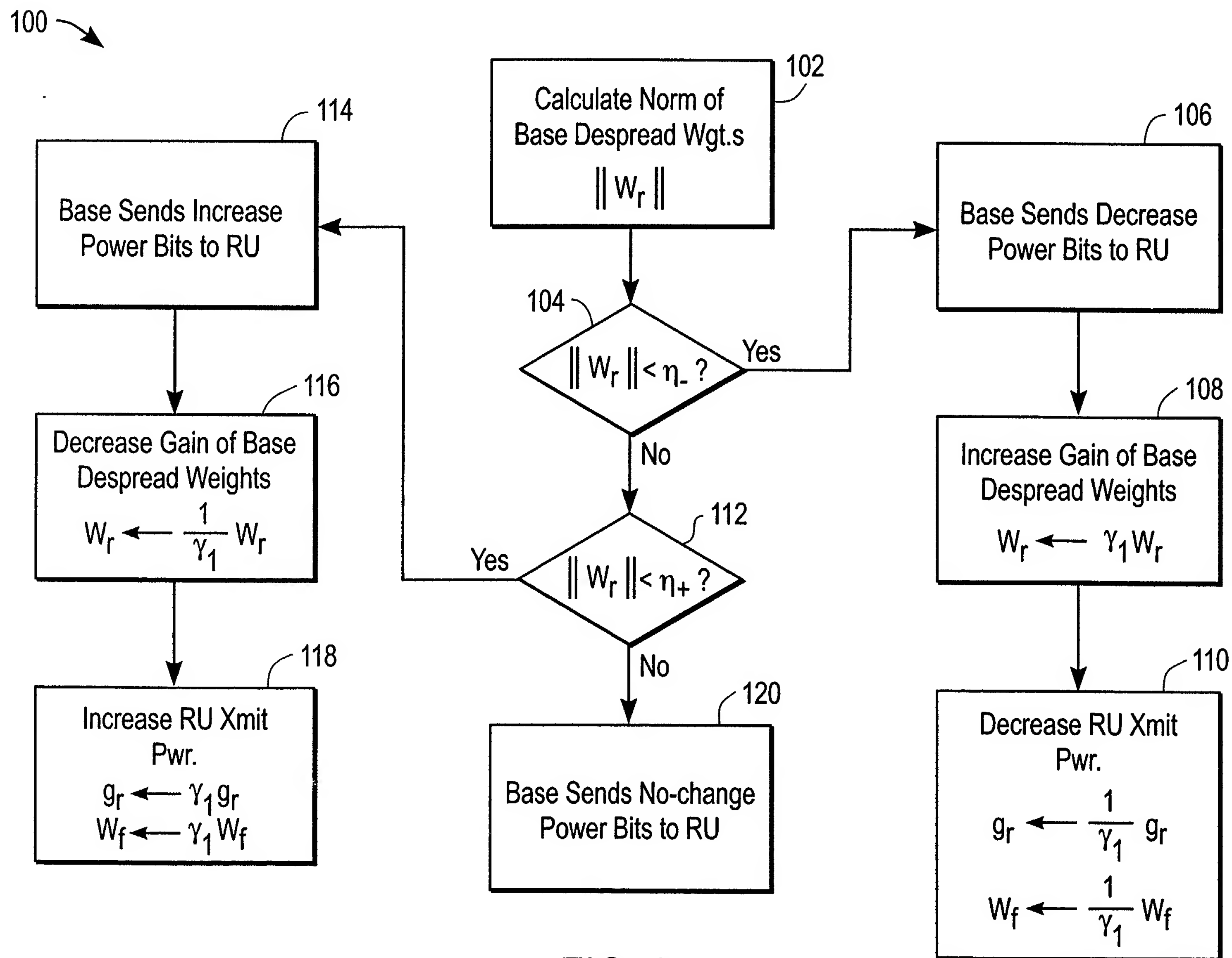

100
Calculate Norm of Base Despread Wgt.s $\|W_r\|$
102
$\|W_r\| < \eta_-$?
104
Yes
No
Base Sends Decrease Power Bits to RU
106
Increase Gain of Base Despread Weights $W_r \leftarrow \gamma_1 W_r$
108
Decrease RU Xmit Pwr. $g_r \leftarrow \frac{1}{\gamma_1} g_r$ $W_f \leftarrow \frac{1}{\gamma_1} W_f$
110
$\|W_r\| < \eta_+$?
112
Yes
No
Base Sends Increase Power Bits to RU
114
Decrease Gain of Base Despread Weights $W_r \leftarrow \frac{1}{\gamma_1} W_r$
116
Increase RU Xmit Pwr. $g_r \leftarrow \gamma_1 g_r$ $W_f \leftarrow \gamma_1 W_f$
118
Base Sends No-change Power Bits to RU
120

FIG. 8

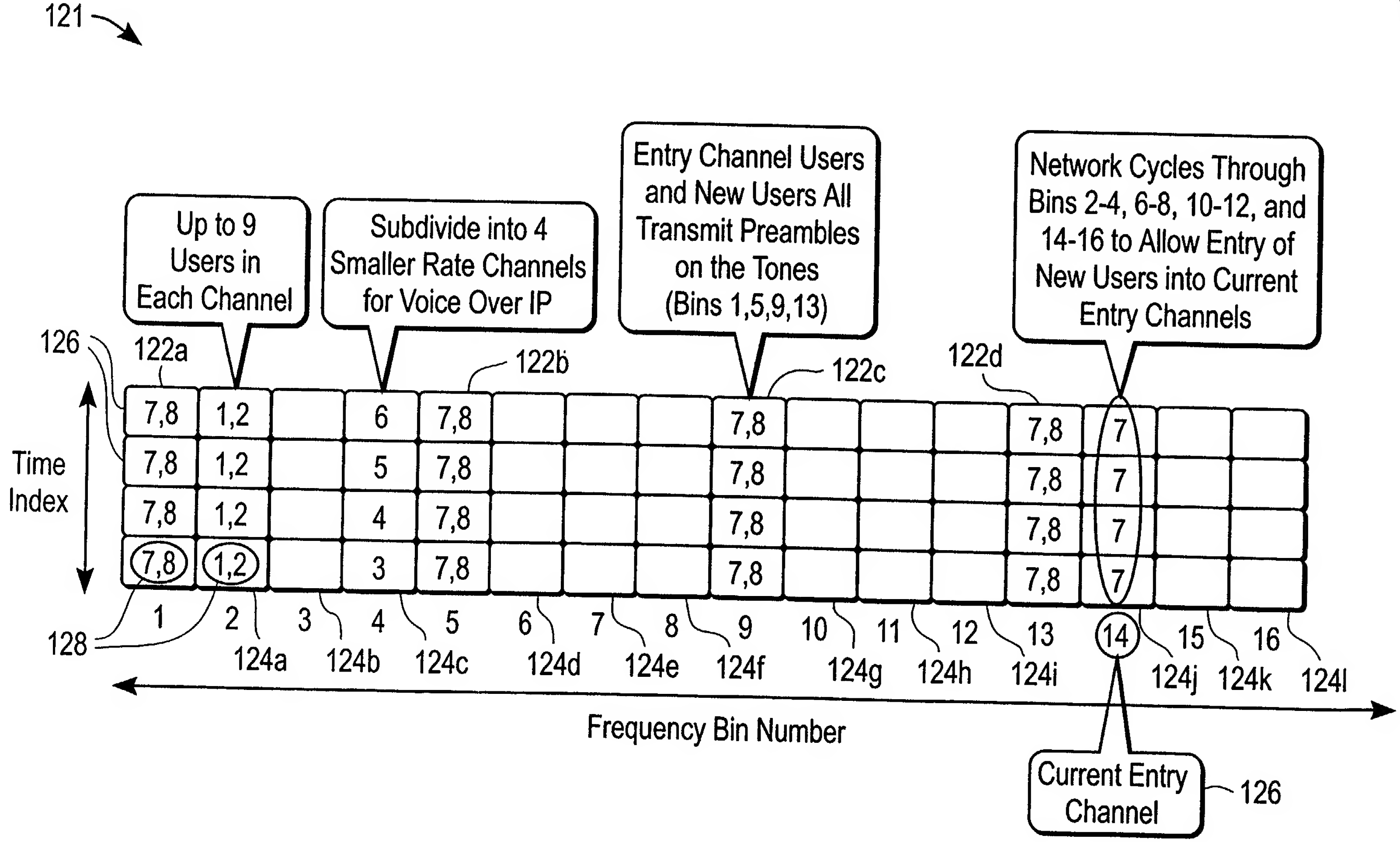

121
Up to 9 Users in Each Channel
Subdivide into 4 Smaller Rate Channels for Voice Over IP
Entry Channel Users and New Users All Transmit Preambles on the Tones (Bins 1,5,9,13)
Network Cycles Through Bins 2-4, 6-8, 10-12, and 14-16 to Allow Entry of New Users into Current Entry Channels
126
122a
122b
122c
122d
Time Index
7,8
1,2
6
5
4
3
7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
128
124a
124b
124c
124d
124e
124f
124g
124h
124i
124j
124k
124l
Frequency Bin Number
Current Entry Channel
126

FIG. 9

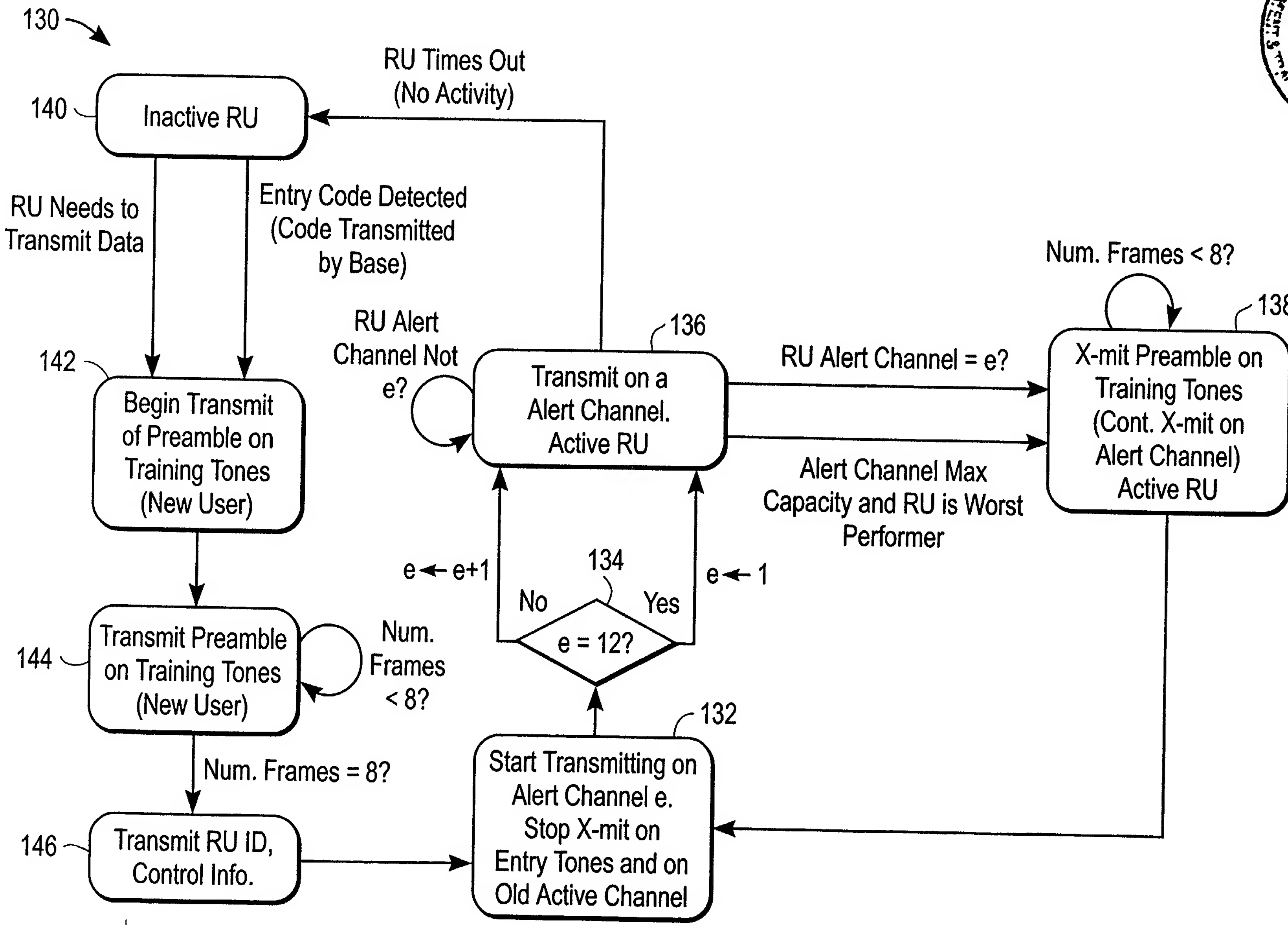

130
140
Inactive RU
RU Needs to Transmit Data
Entry Code Detected (Code Transmitted by Base)
RU Times Out (No Activity)
142
Begin Transmit of Preamble on Training Tones (New User)
144
Transmit Preamble on Training Tones (New User)
Num. Frames < 8?
Num. Frames = 8?
146
Transmit RU ID, Control Info.
136
Transmit on a Alert Channel. Active RU
RU Alert Channel Not e?
RU Alert Channel = e?
Alert Channel Max Capacity and RU is Worst Performer
138
X-mit Preamble on Training Tones (Cont. X-mit on Alert Channel) Active RU
Num. Frames < 8?
134
e = 12?
No
Yes
e ← e+1
e ← 1
132
Start Transmitting on Alert Channel e. Stop X-mit on Entry Tones and on Old Active Channel

FIG. 10

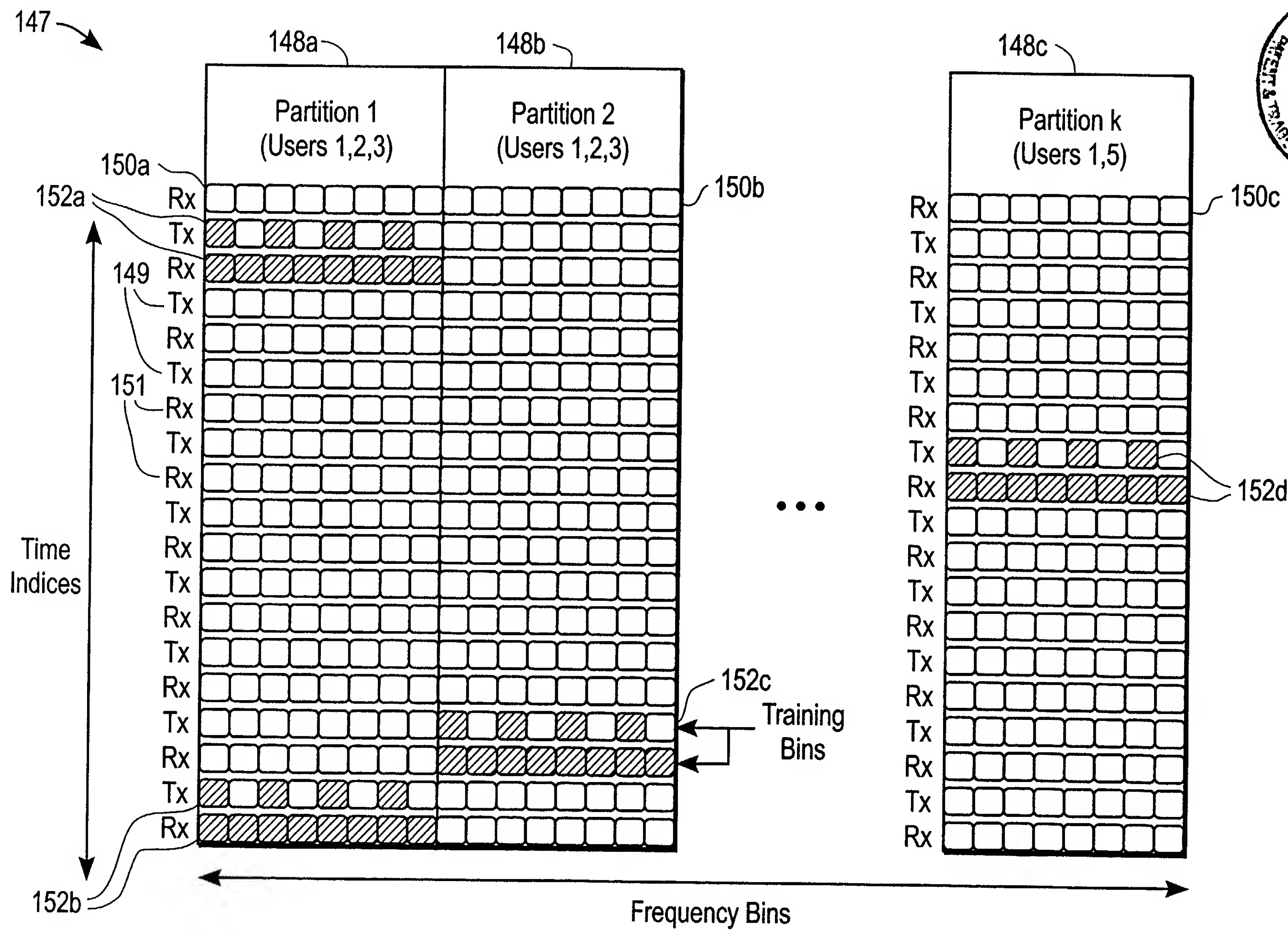

147
148a
148b
148c
Partition 1
(Users 1,2,3)
Partition 2
(Users 1,2,3)
Partition k
(Users 1,5)
150a
150b
150c
152a
152b
152c
152d
149
151
Rx
Tx
Training
Bins
Time
Indices
Frequency Bins

FIG. 11

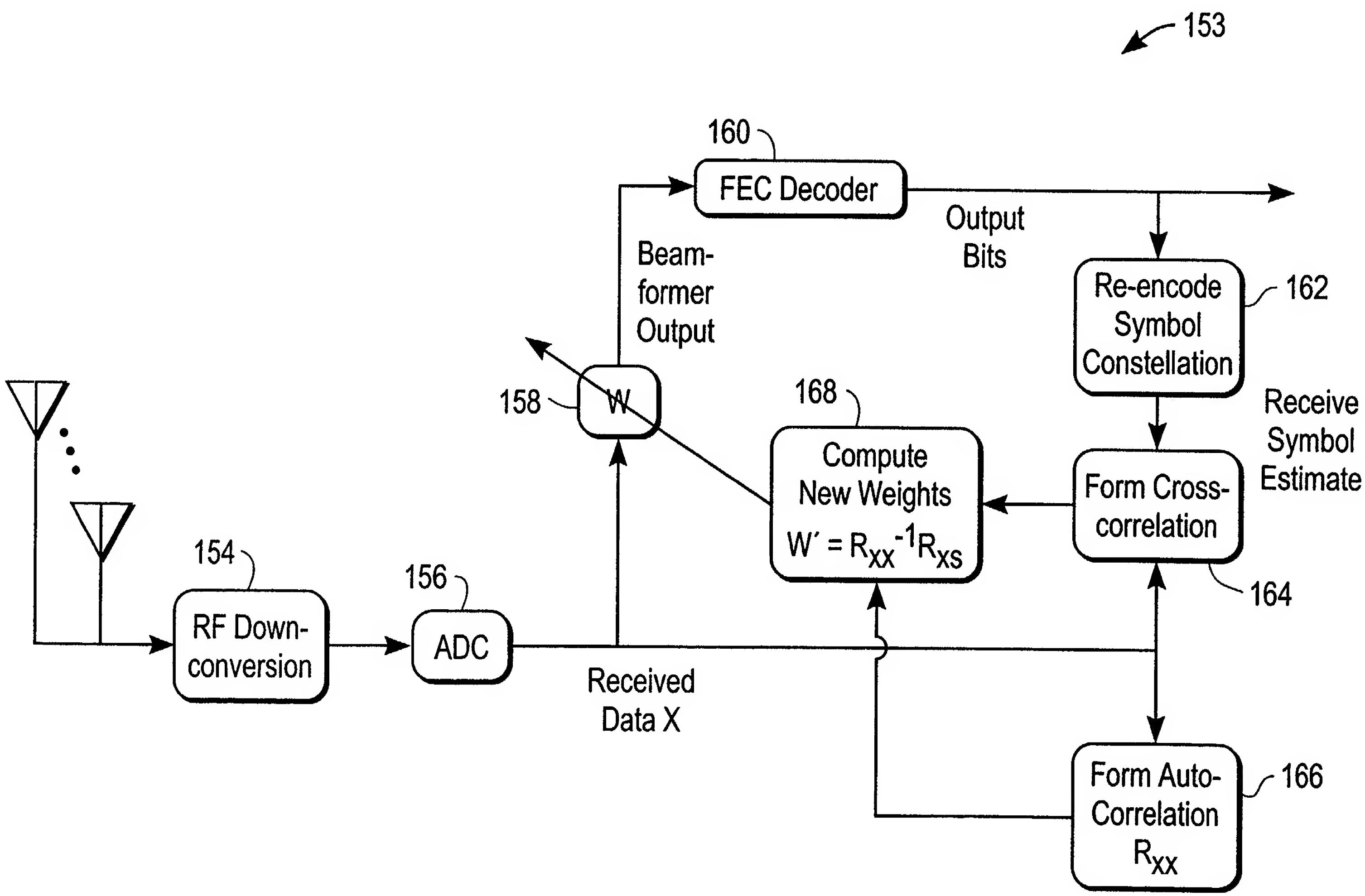

153
160
FEC Decoder
Output
Bits
Beam-
former
Output
Re-encode
Symbol
Constellation
162
158
W
168
Compute
New Weights
W′ = $R_{XX}^{-1}R_{XS}$
Receive
Symbol
Estimate
Form Cross-
correlation
164
154
RF Down-
conversion
156
ADC
Received
Data X
Form Auto-
Correlation
R_{XX}
166

FIG. 12

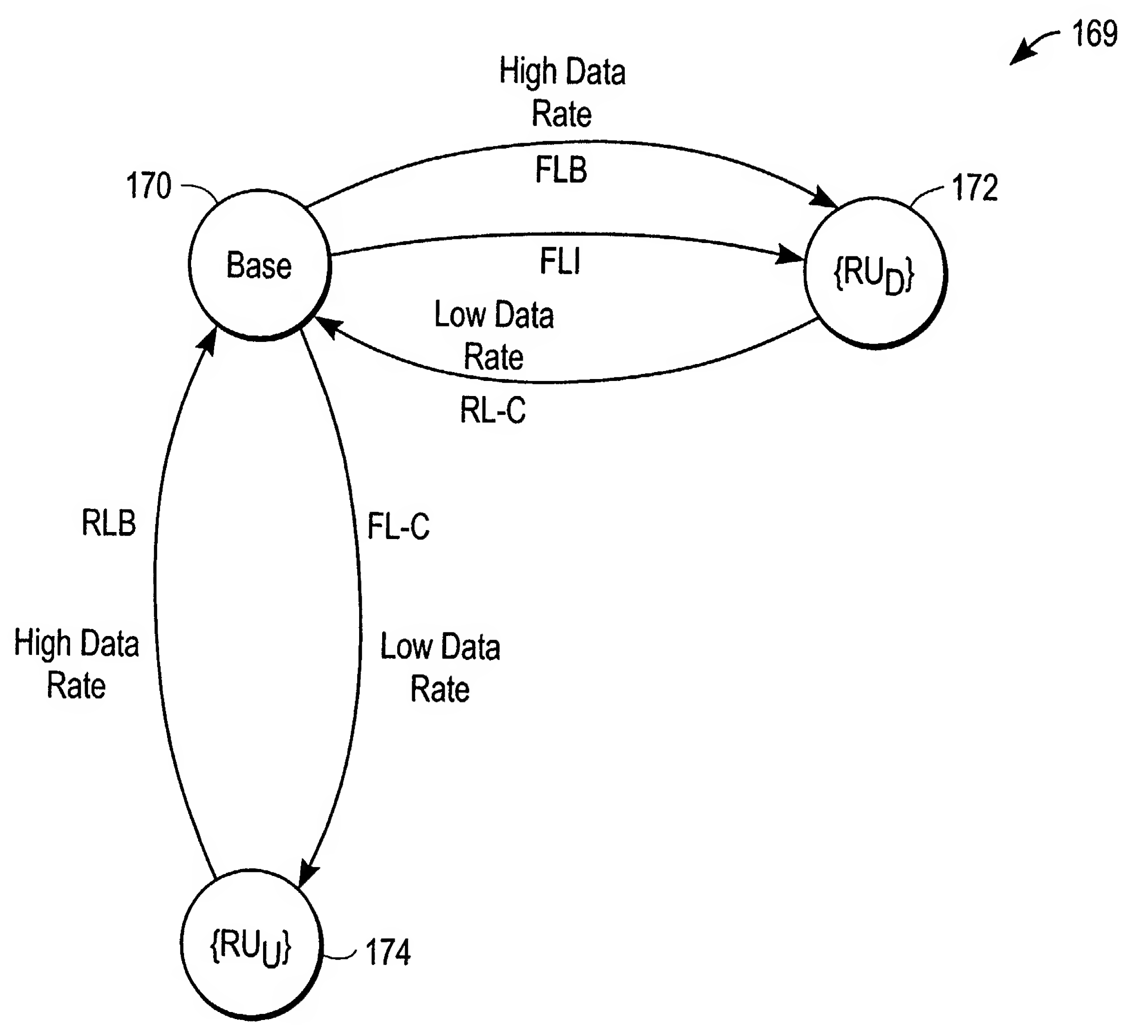

169
High Data
Rate
FLB
FLI
Low Data
Rate
RL-C
170
Base
172
{RU_D}
RLB
High Data
Rate
FL-C
Low Data
Rate
{RU_U}
174

FIG. 13

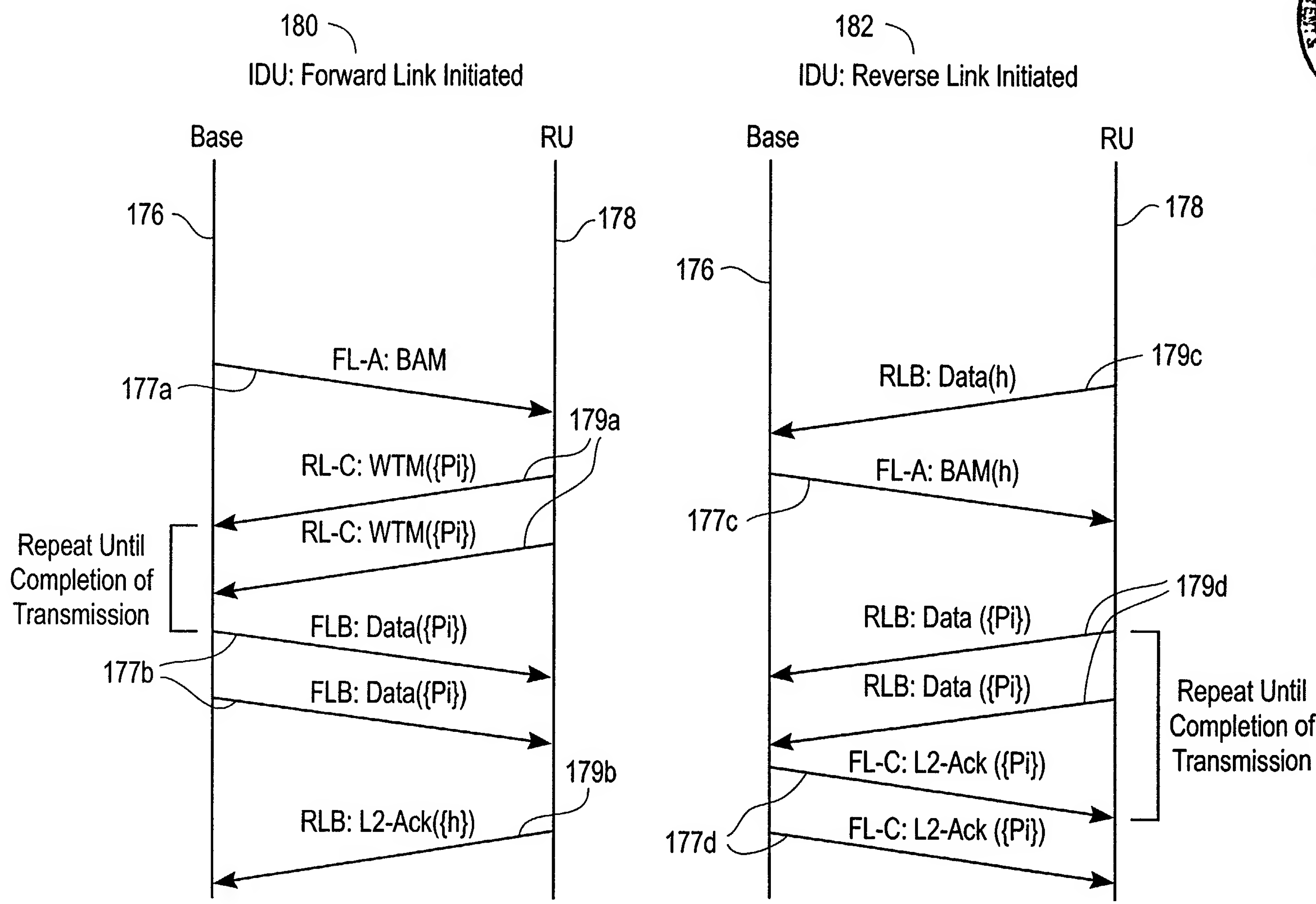

180
IDU: Forward Link Initiated
Base
RU
176
178
FL-A: BAM
177a
179a
RL-C: WTM({Pi})
RL-C: WTM({Pi})
Repeat Until
Completion of
Transmission
FLB: Data({Pi})
177b
FLB: Data({Pi})
179b
RLB: L2-Ack({h})
182
IDU: Reverse Link Initiated
Base
RU
178
176
179c
RLB: Data(h)
FL-A: BAM(h)
177c
179d
RLB: Data ({Pi})
RLB: Data ({Pi})
Repeat Until
Completion of
Transmission
FL-C: L2-Ack ({Pi})
177d
FL-C: L2-Ack ({Pi})

FIG. 14

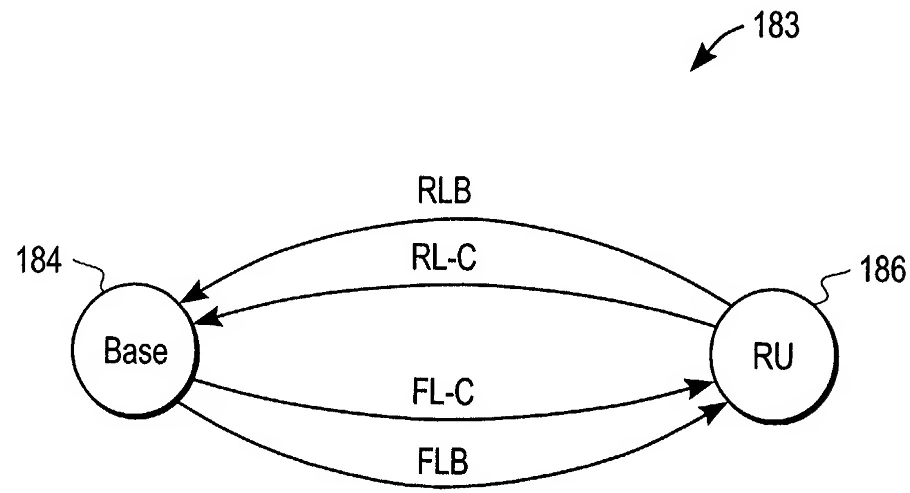


FIG. 15

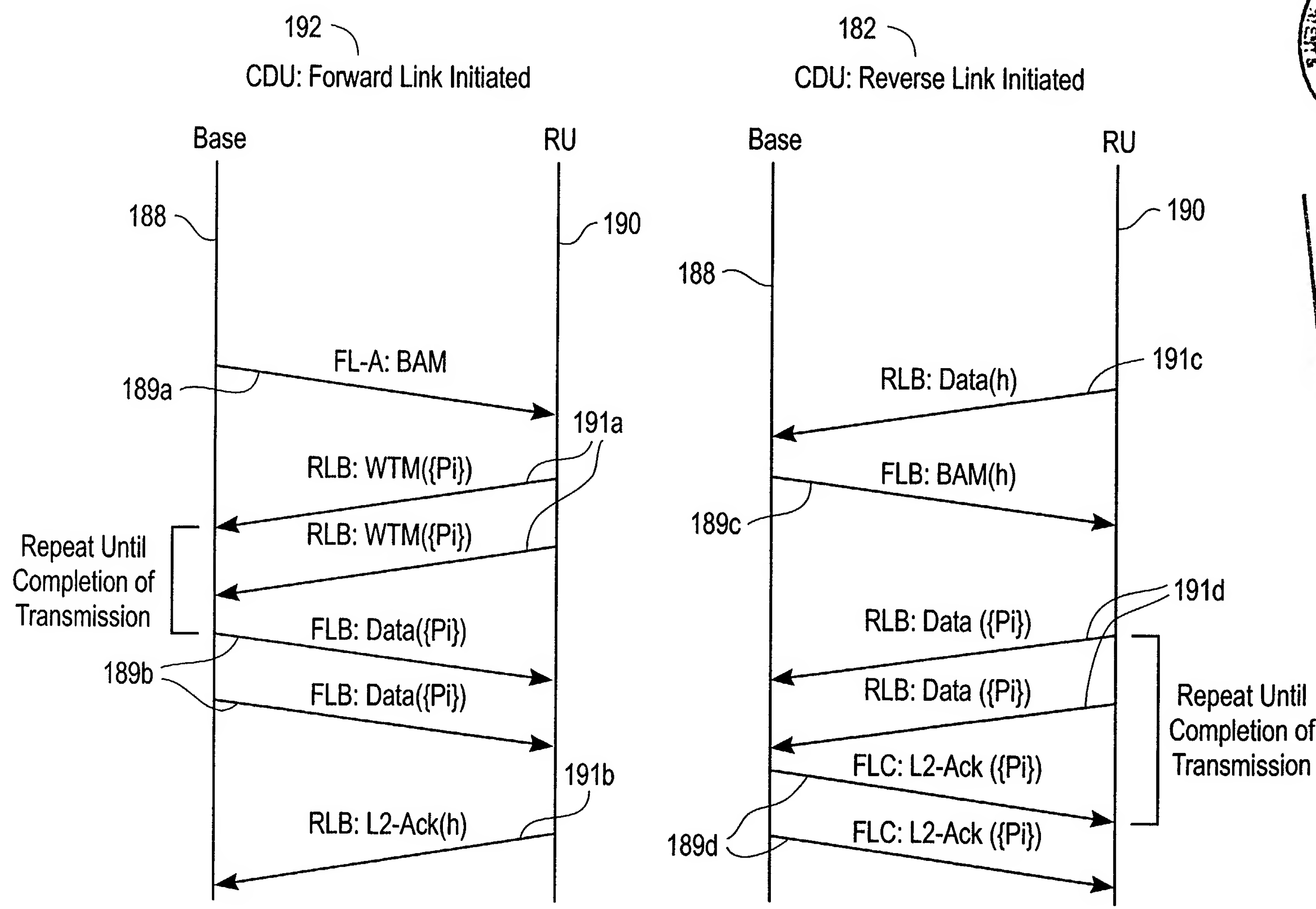

192
CDU: Forward Link Initiated
Base
RU
188
190
FL-A: BAM
189a
191a
RLB: WTM({Pi})
RLB: WTM({Pi})
Repeat Until Completion of Transmission
FLB: Data({Pi})
FLB: Data({Pi})
189b
191b
RLB: L2-Ack(h)
182
CDU: Reverse Link Initiated
Base
RU
190
188
191c
RLB: Data(h)
FLB: BAM(h)
189c
191d
RLB: Data ({Pi})
RLB: Data ({Pi})
FLC: L2-Ack ({Pi})
FLC: L2-Ack ({Pi})
189d
Repeat Until Completion of Transmission

FIG. 16

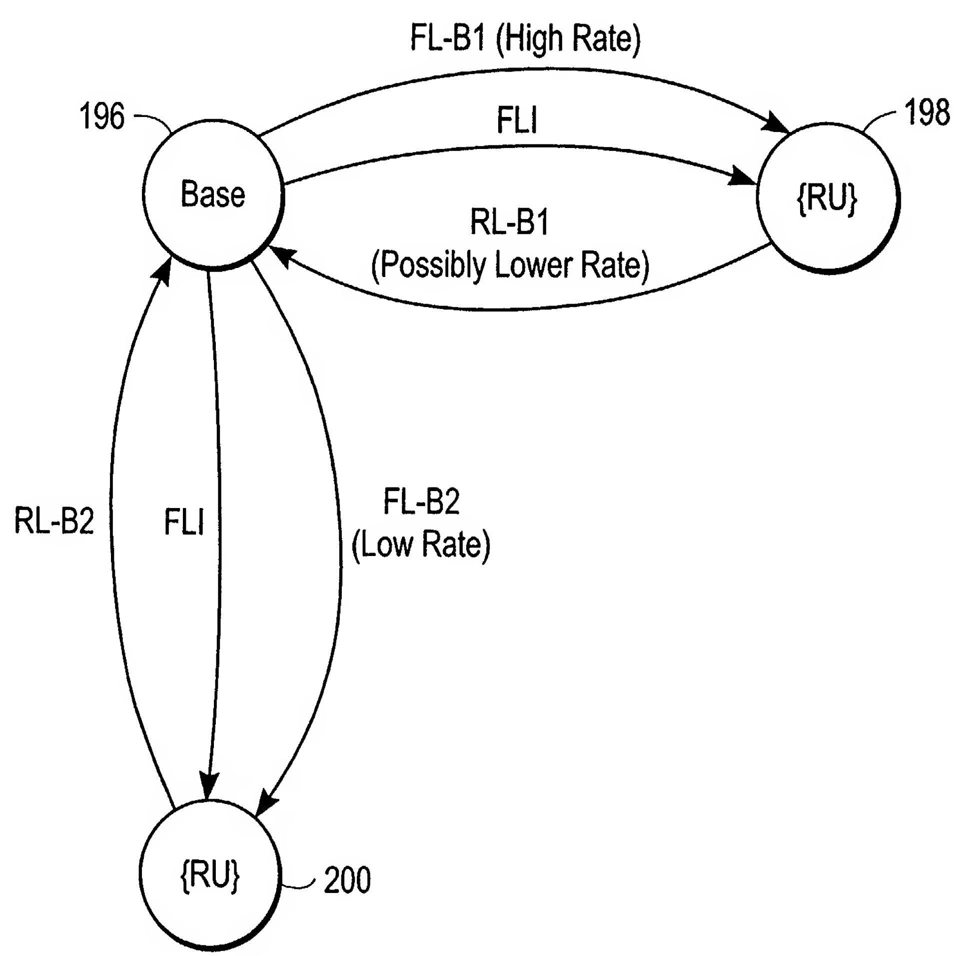

FL-B1 (High Rate)
FLI
RL-B1
(Possibly Lower Rate)
196
Base
198
{RU}
RL-B2
FLI
FL-B2
(Low Rate)
{RU}
200

FIG. 17

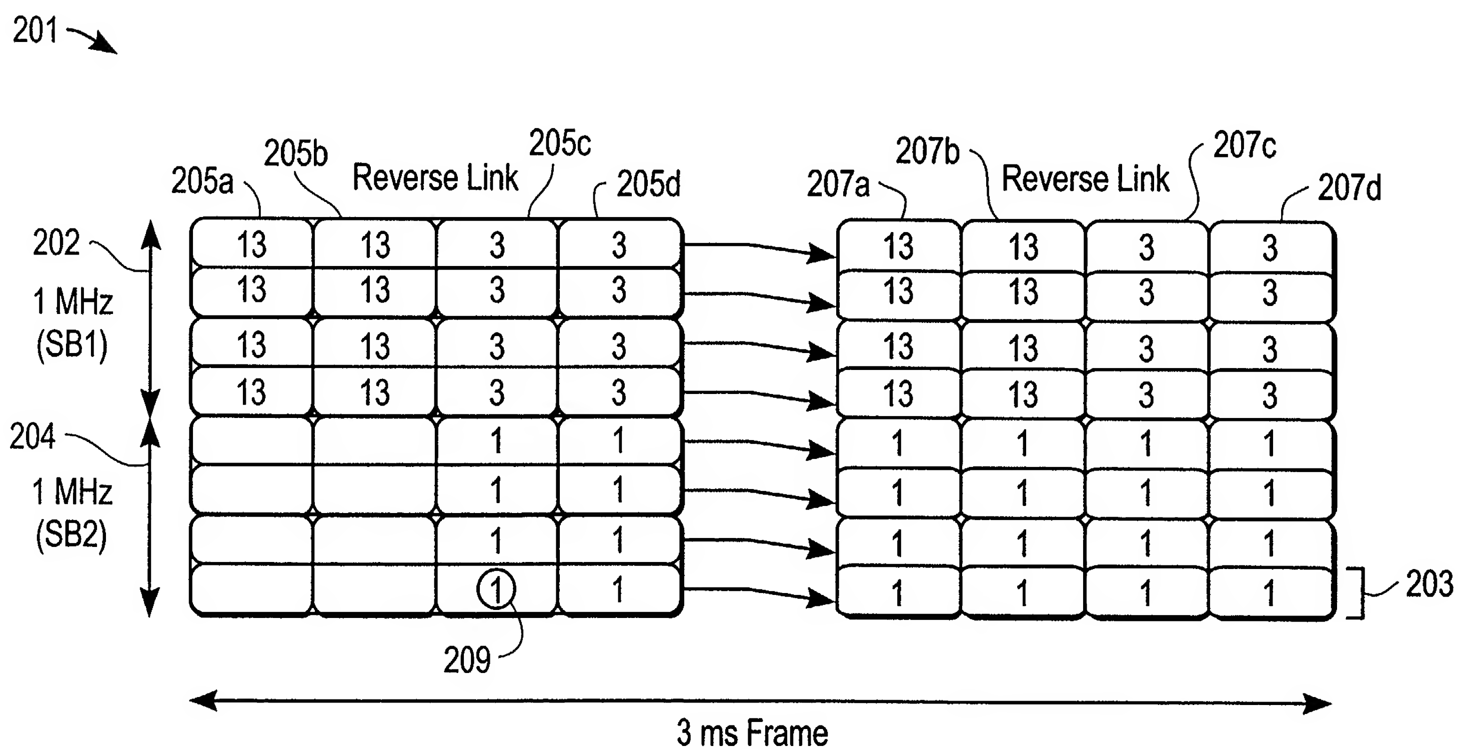

201
202
1 MHz (SB1)
204
1 MHz (SB2)
205a
205b
Reverse Link
205c
205d
207a
207b
Reverse Link
207c
207d
13
3
1
209
203
3 ms Frame

FIG. 18

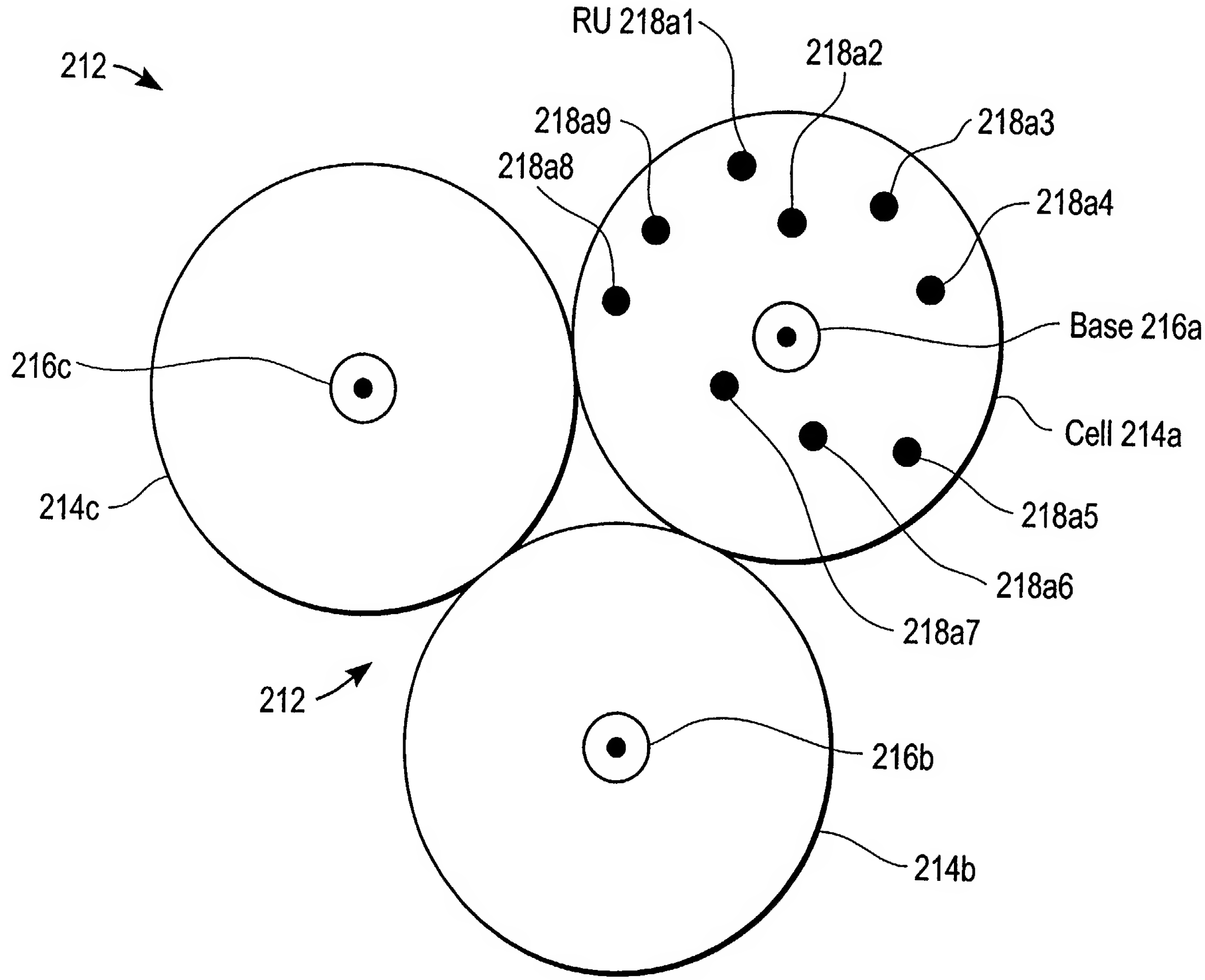

212
RU 218a1
218a2
218a3
218a4
218a5
218a6
218a7
218a8
218a9
Base 216a
Cell 214a
216b
214b
216c
214c
212

FIG. 19

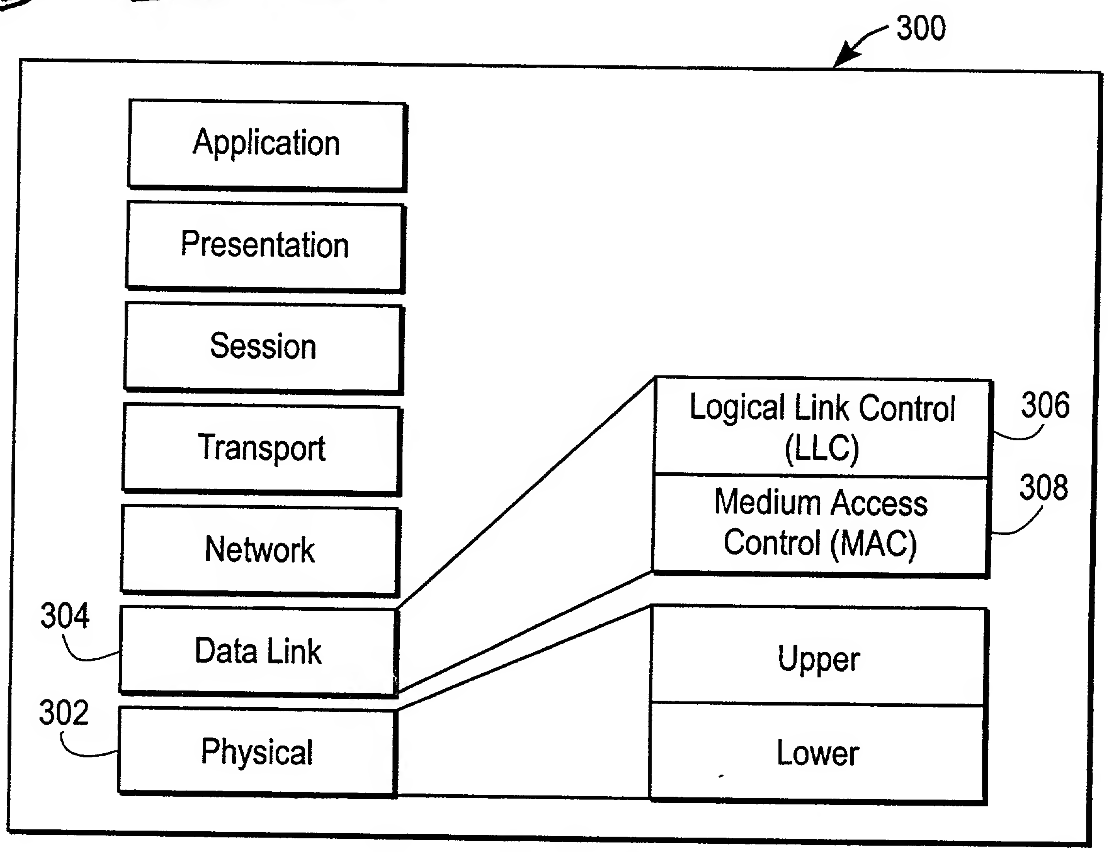


FIG. 20
(PRIOR ART)

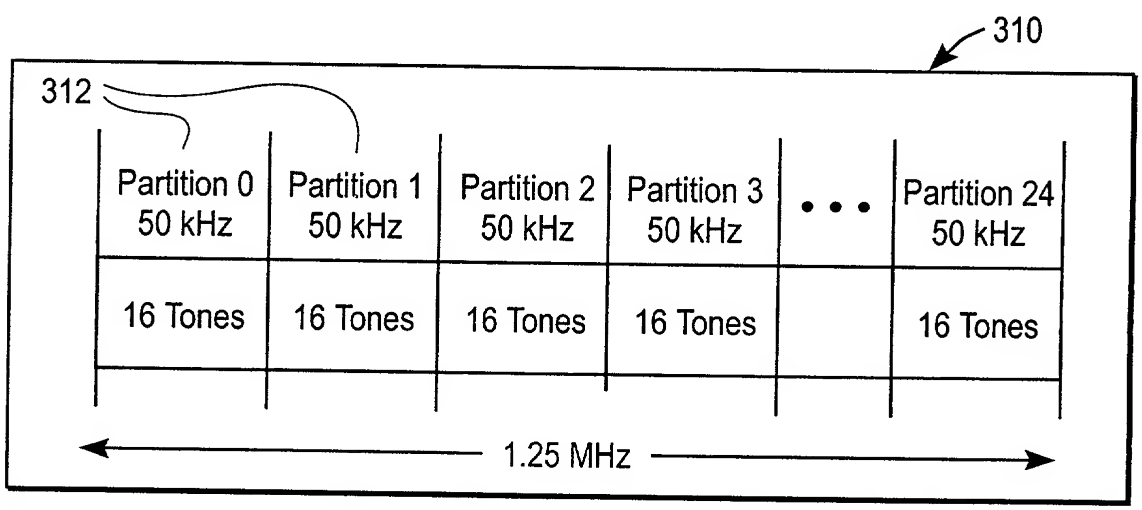


FIG. 21

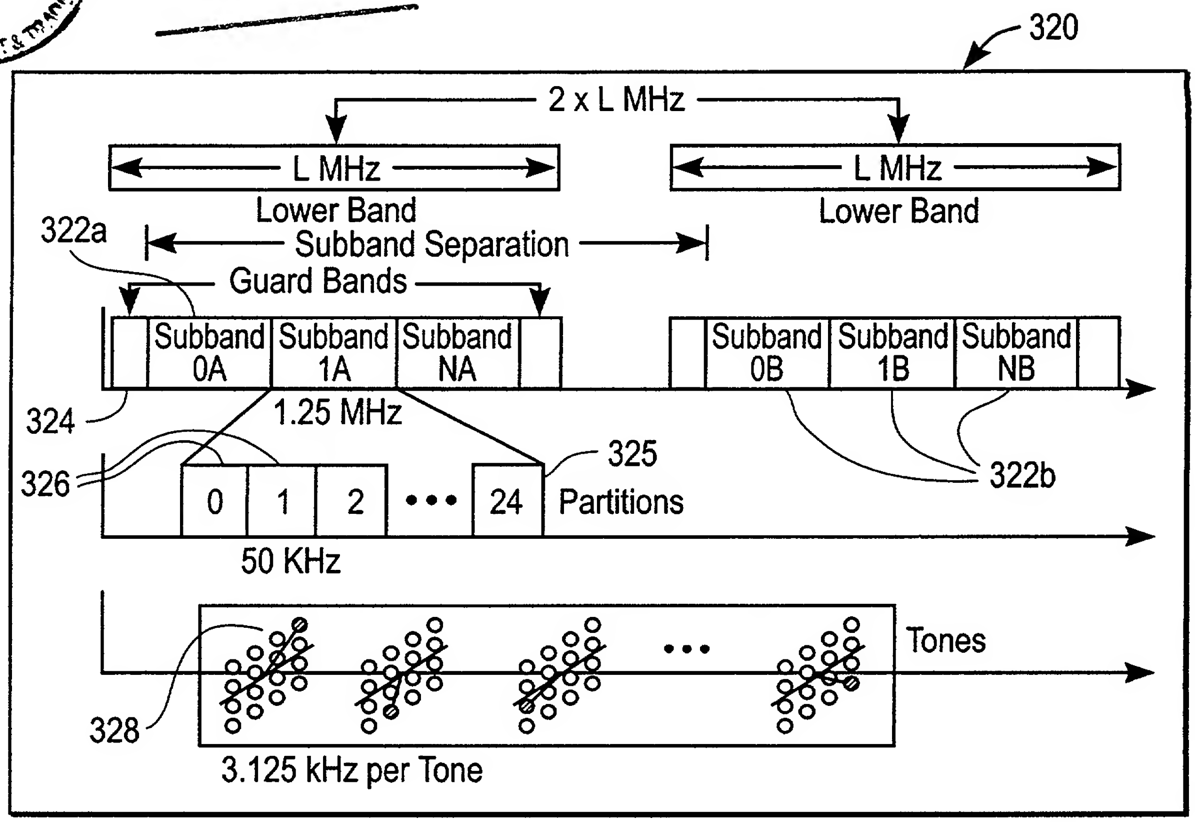

320
2 x L MHz
L MHz
Lower Band
L MHz
Lower Band
322a
Subband Separation
Guard Bands
Subband 0A
Subband 1A
Subband NA
Subband 0B
Subband 1B
Subband NB
324
1.25 MHz
325
326
0
1
2
24
Partitions
322b
50 KHz
Tones
328
3.125 kHz per Tone

FIG. 22

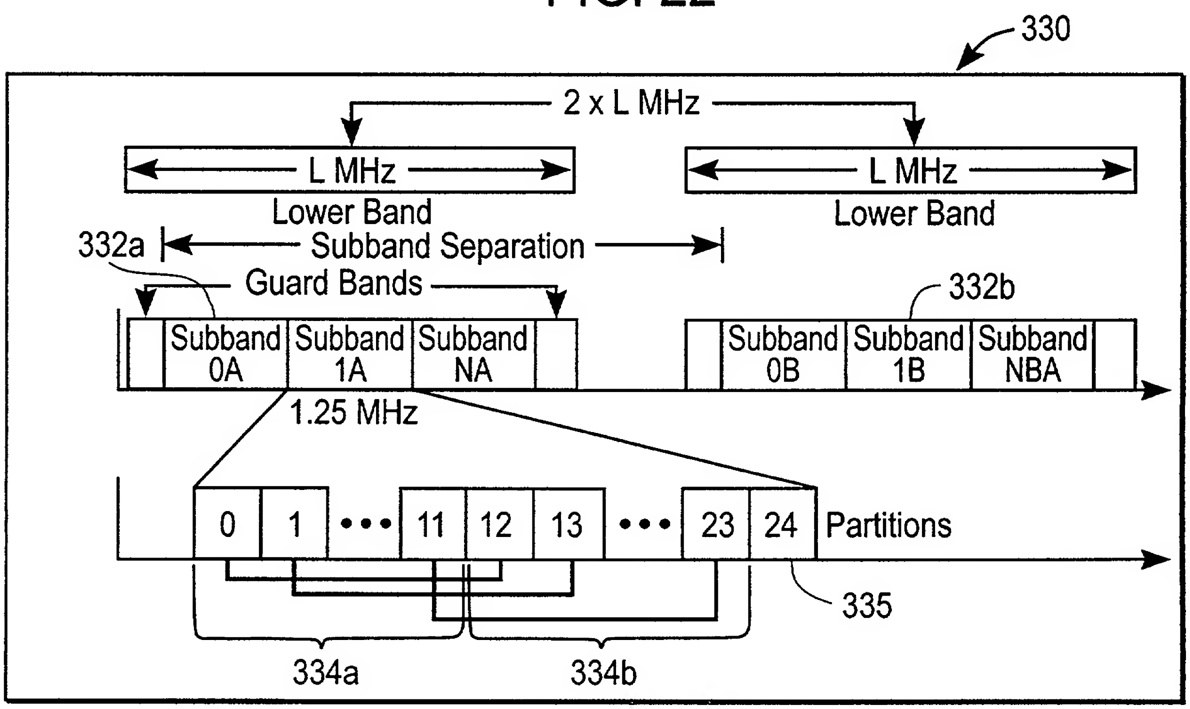

330
2 x L MHz
L MHz
Lower Band
L MHz
Lower Band
332a
Subband Separation
Guard Bands
332b
Subband 0A
Subband 1A
Subband NA
Subband 0B
Subband 1B
Subband NBA
1.25 MHz
0
1
11
12
13
23
24
Partitions
335
334a
334b

FIG. 23

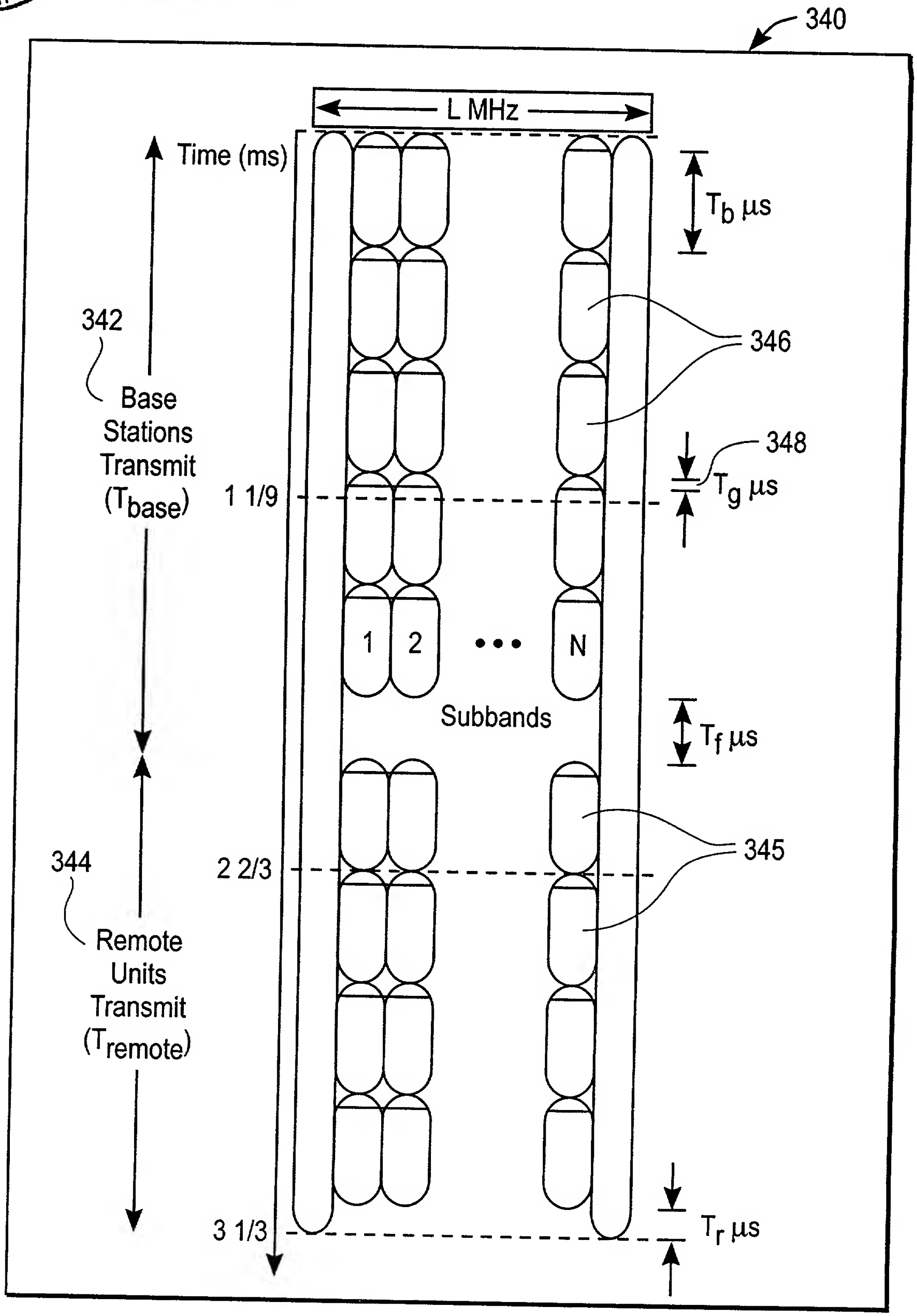

340
L MHz
Time (ms)
342
Base Stations Transmit (T_{base})
344
Remote Units Transmit (T_{remote})
1 1/9
2 2/3
3 1/3
T_b µs
346
348
T_g µs
1
2
N
Subbands
T_f µs
345
T_r µs

FIG. 24

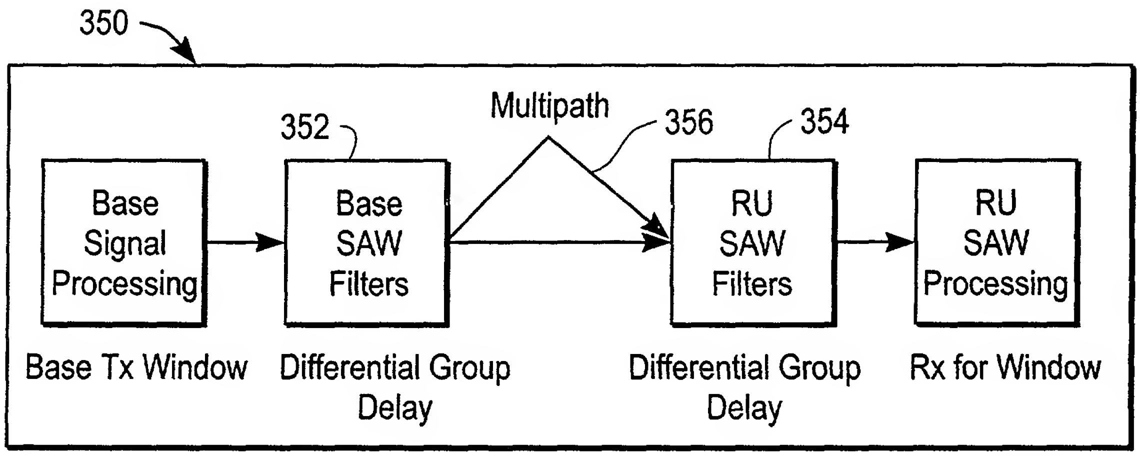

350
352
Multipath
356
354
Base Signal Processing
Base SAW Filters
RU SAW Filters
RU SAW Processing
Base Tx Window
Differential Group Delay
Differential Group Delay
Rx for Window

FIG. 25

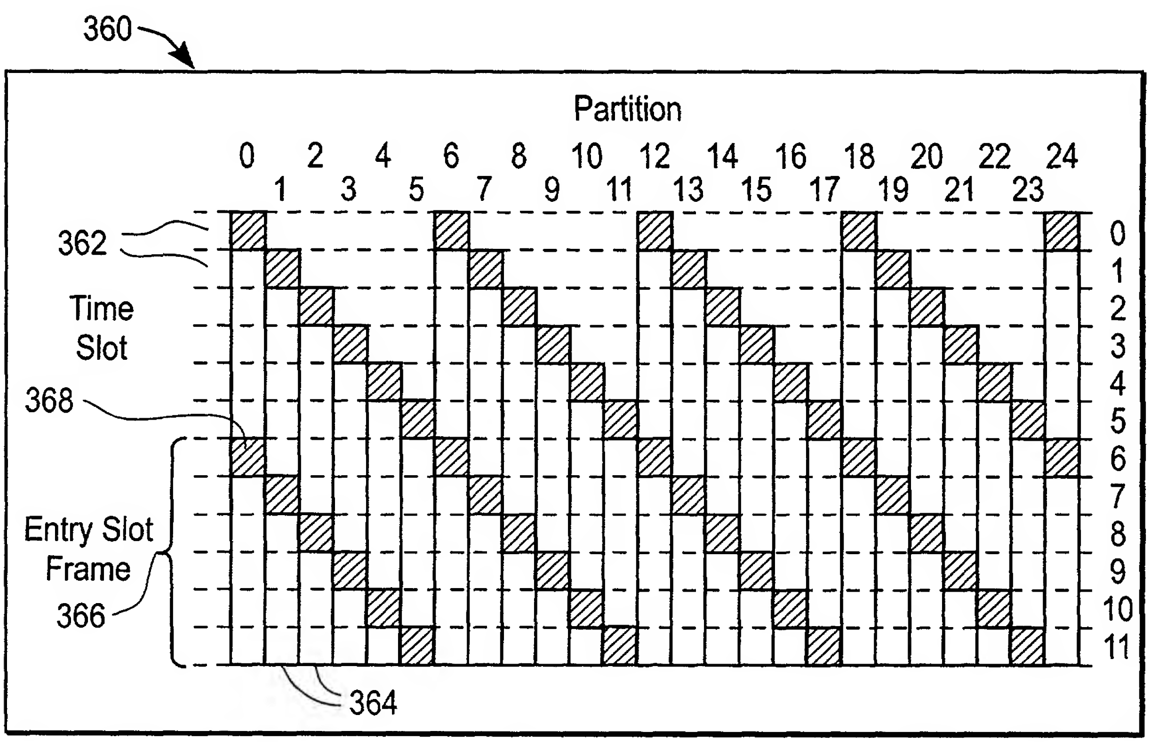

360
Partition
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
362
Time Slot
368
Entry Slot Frame
366
364
0 1 2 3 4 5 6 7 8 9 10 11

FIG. 26

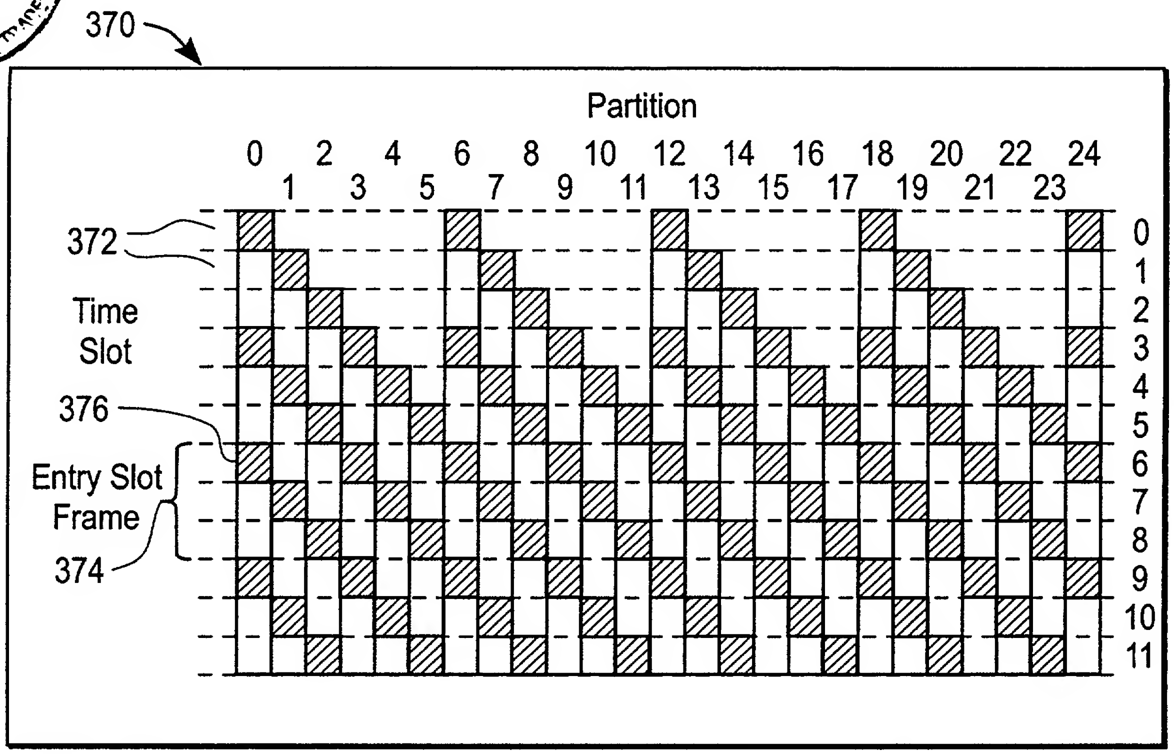

370
Partition
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
372
Time
Slot
376
Entry Slot
Frame
374
0
1
2
3
4
5
6
7
8
9
10
11

FIG. 27

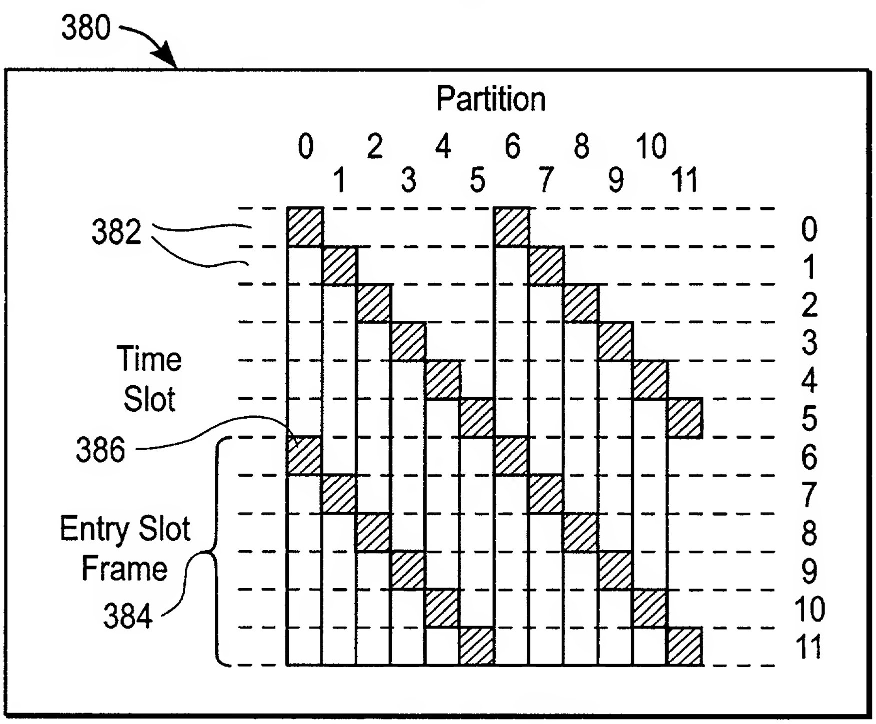

380
Partition
0 1 2 3 4 5 6 7 8 9 10 11
382
Time
Slot
386
Entry Slot
Frame
384
0
1
2
3
4
5
6
7
8
9
10
11

FIG. 28

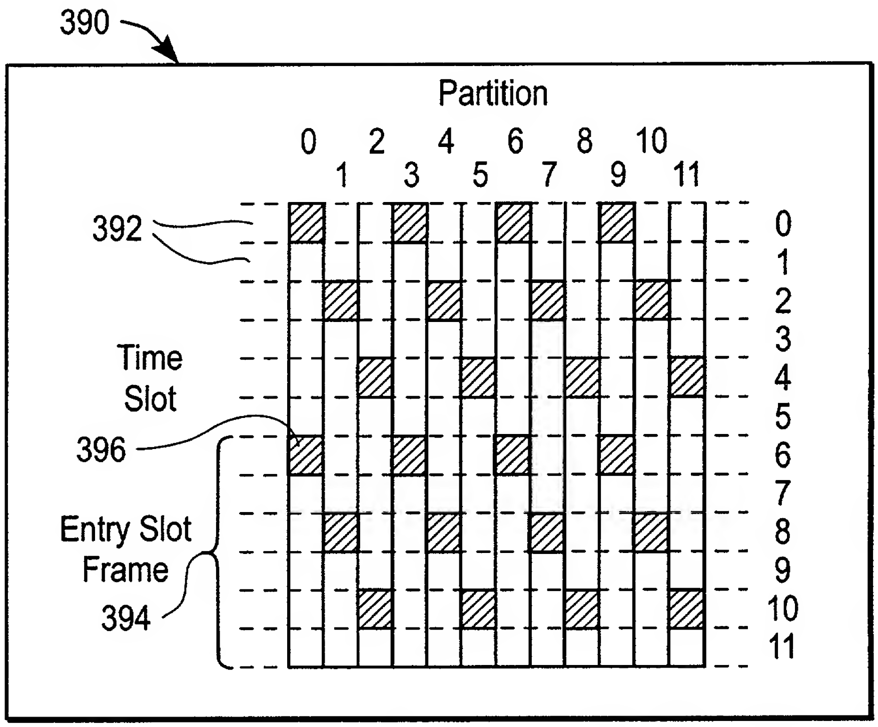

390
Partition
0 1 2 3 4 5 6 7 8 9 10 11
392
Time
Slot
396
Entry Slot
Frame
394
0 1 2 3 4 5 6 7 8 9 10 11

FIG. 29

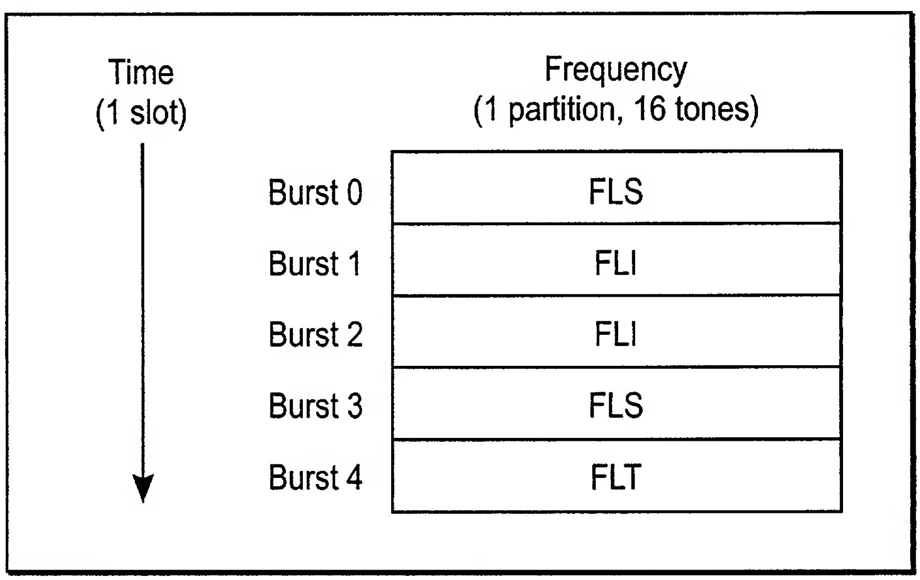

Time
(1 slot)
Frequency
(1 partition, 16 tones)
Burst 0
FLS
Burst 1
FLI
Burst 2
FLI
Burst 3
FLS
Burst 4
FLT

FIG. 30

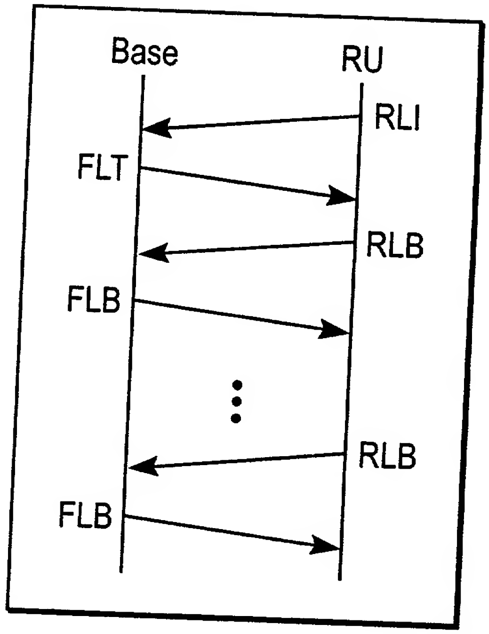

Base
RU
RLI
FLT
RLB
FLB
RLB
FLB

FIG. 31

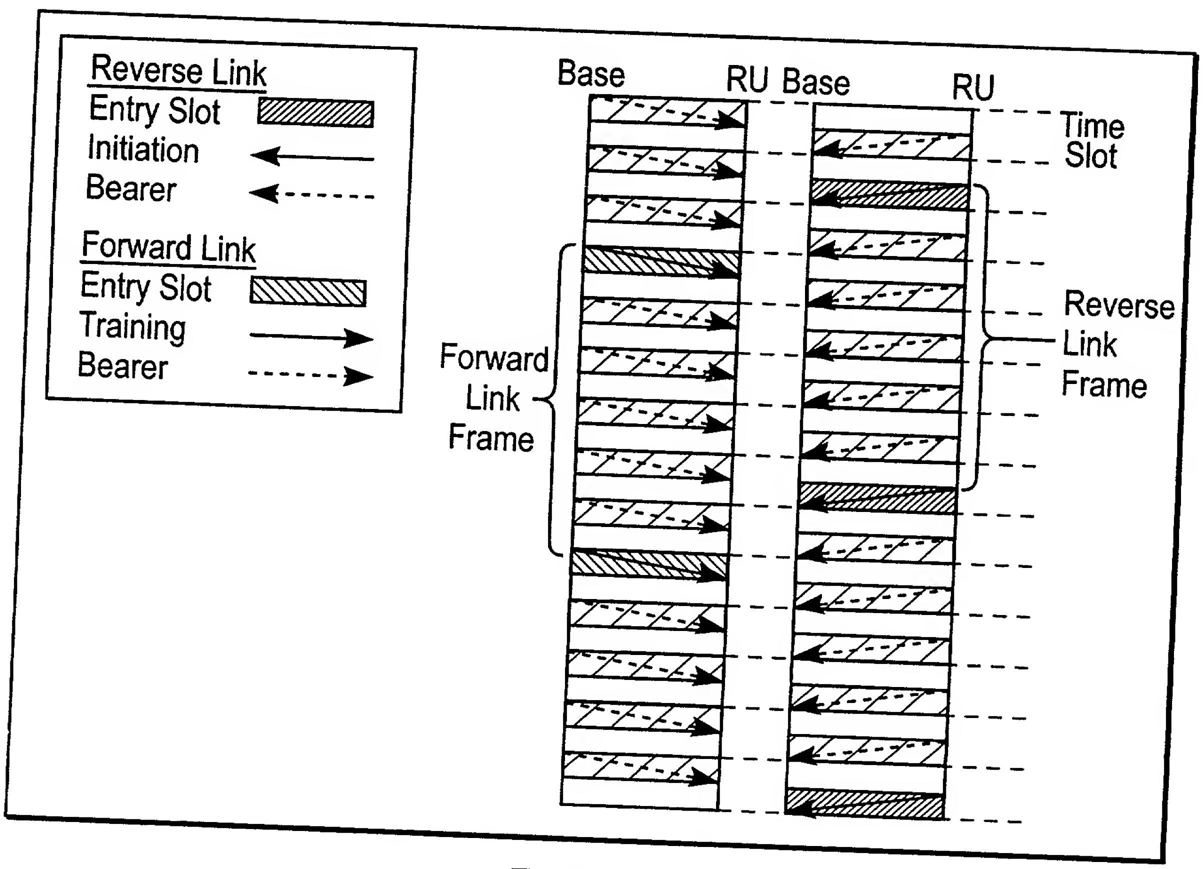

Reverse Link
Entry Slot
Initiation
Bearer
Forward Link
Entry Slot
Training
Bearer
Base
RU
Base
RU
Time Slot
Reverse Link Frame
Forward Link Frame

FIG. 32

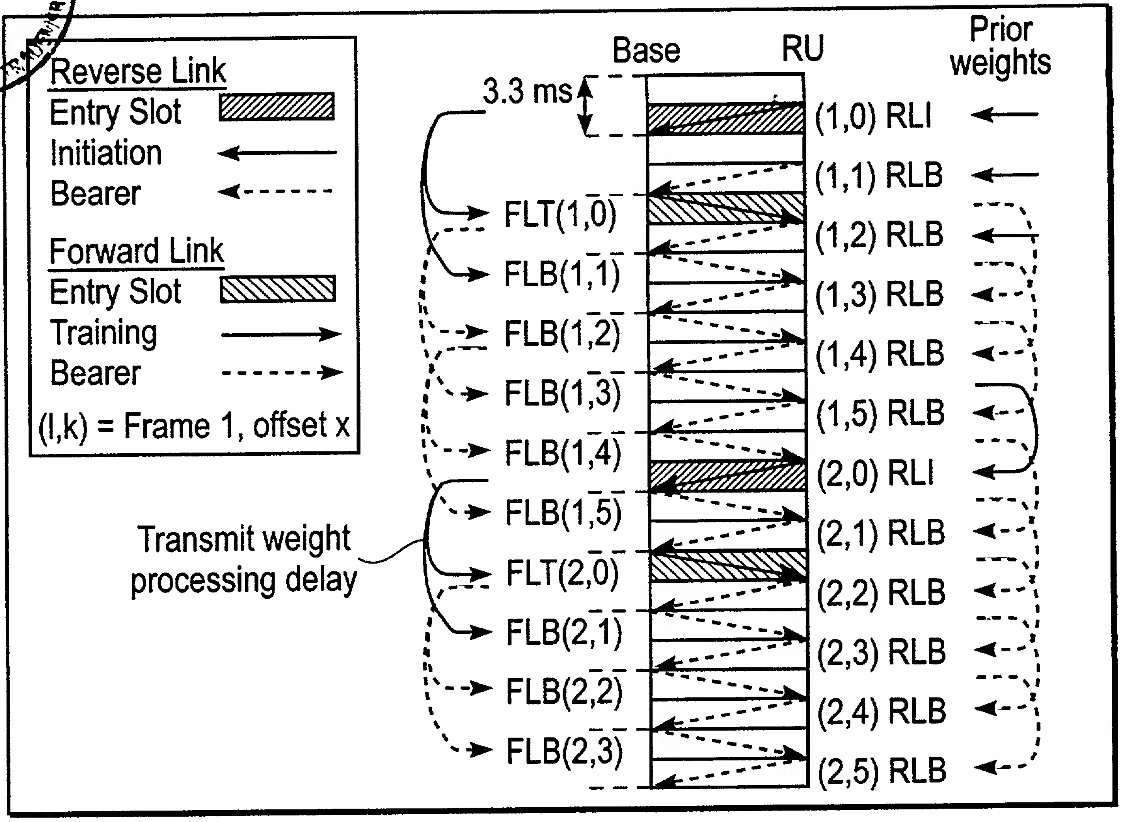

Reverse Link
Entry Slot
Initiation
Bearer
Forward Link
Entry Slot
Training
Bearer
(l,k) = Frame 1, offset x
Base
RU
Prior weights
3.3 ms
FLT(1,0)
FLB(1,1)
FLB(1,2)
FLB(1,3)
FLB(1,4)
FLB(1,5)
FLT(2,0)
FLB(2,1)
FLB(2,2)
FLB(2,3)
Transmit weight processing delay
(1,0) RLI
(1,1) RLB
(1,2) RLB
(1,3) RLB
(1,4) RLB
(1,5) RLB
(2,0) RLI
(2,1) RLB
(2,2) RLB
(2,3) RLB
(2,4) RLB
(2,5) RLB

FIG. 33

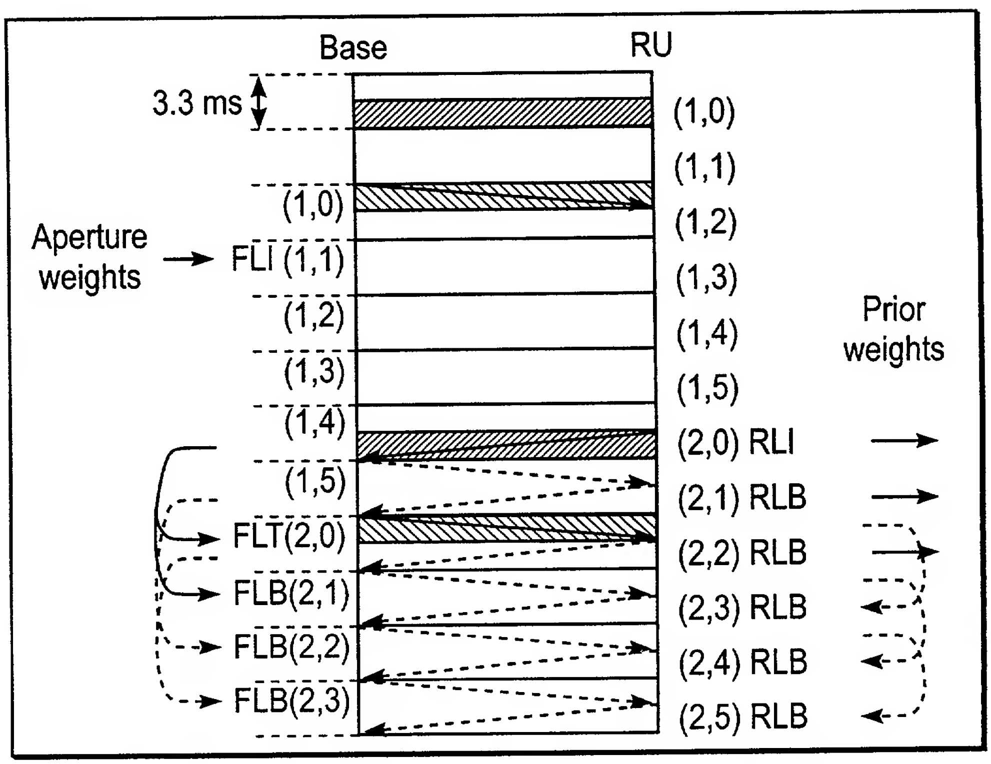

Base
RU
3.3 ms
(1,0)
FLI (1,1)
(1,2)
(1,3)
(1,4)
(1,5)
FLT(2,0)
FLB(2,1)
FLB(2,2)
FLB(2,3)
Aperture weights
Prior weights
(1,0)
(1,1)
(1,2)
(1,3)
(1,4)
(1,5)
(2,0) RLI
(2,1) RLB
(2,2) RLB
(2,3) RLB
(2,4) RLB
(2,5) RLB

FIG. 34

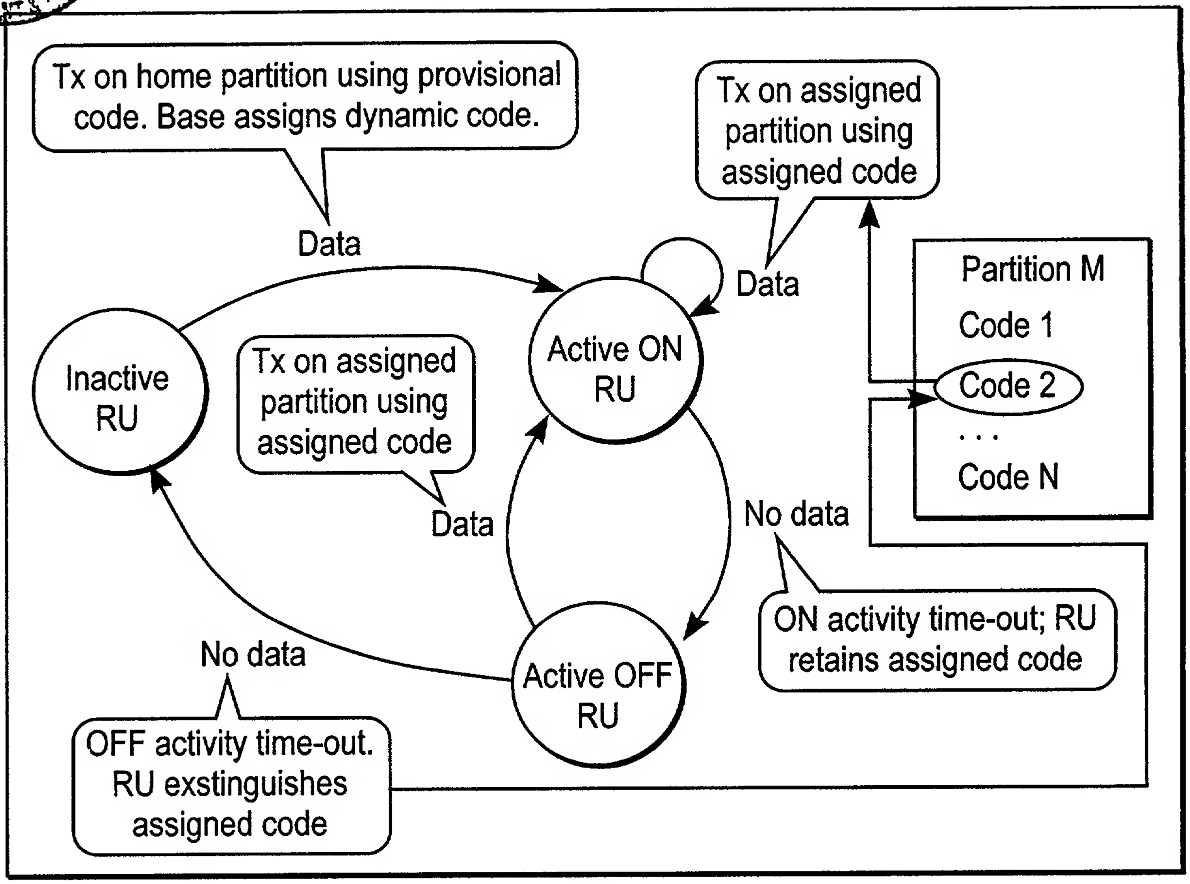

Tx on home partition using provisional code. Base assigns dynamic code.
Data
Tx on assigned partition using assigned code
Data
Partition M
Code 1
Code 2
. . .
Code N
Inactive RU
Tx on assigned partition using assigned code
Active ON RU
Data
No data
ON activity time-out; RU retains assigned code
No data
Active OFF RU
OFF activity time-out. RU exstinguishes assigned code

FIG. 35

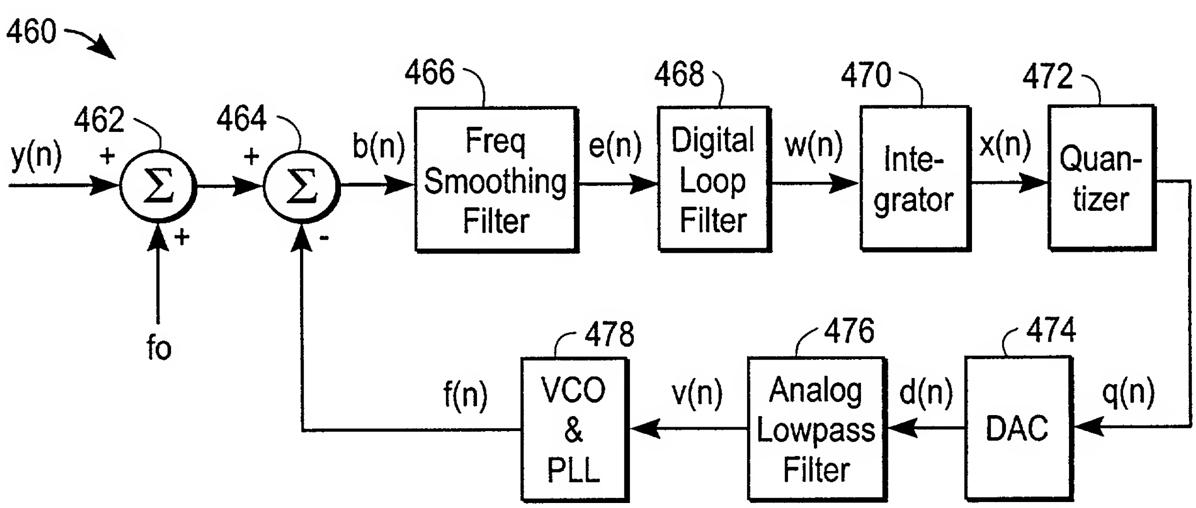

460
462
464
466
468
470
472
y(n)
+
+
+
-
fo
b(n)
Freq Smoothing Filter
e(n)
Digital Loop Filter
w(n)
Inte-grator
x(n)
Quan-tizer
478
476
474
f(n)
VCO & PLL
v(n)
Analog Lowpass Filter
d(n)
DAC
q(n)

FIG. 36

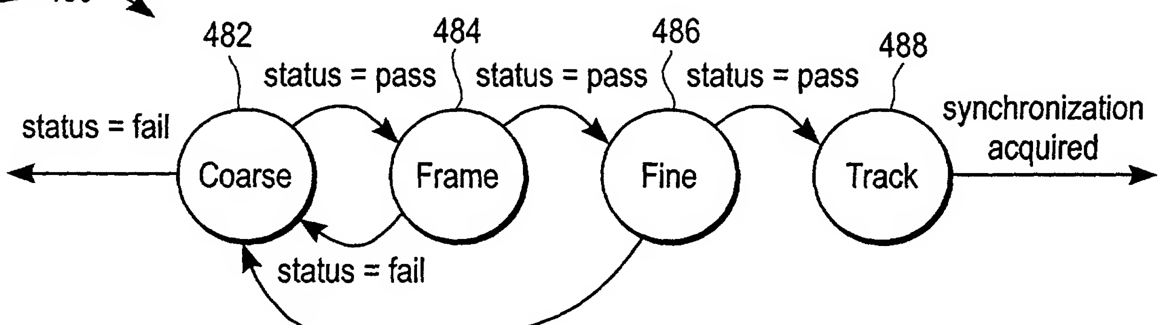

480
482
484
486
488
status = pass
status = pass
status = pass
status = fail
Coarse
Frame
Fine
Track
synchronization acquired
status = fail
status = fail

FIG. 37

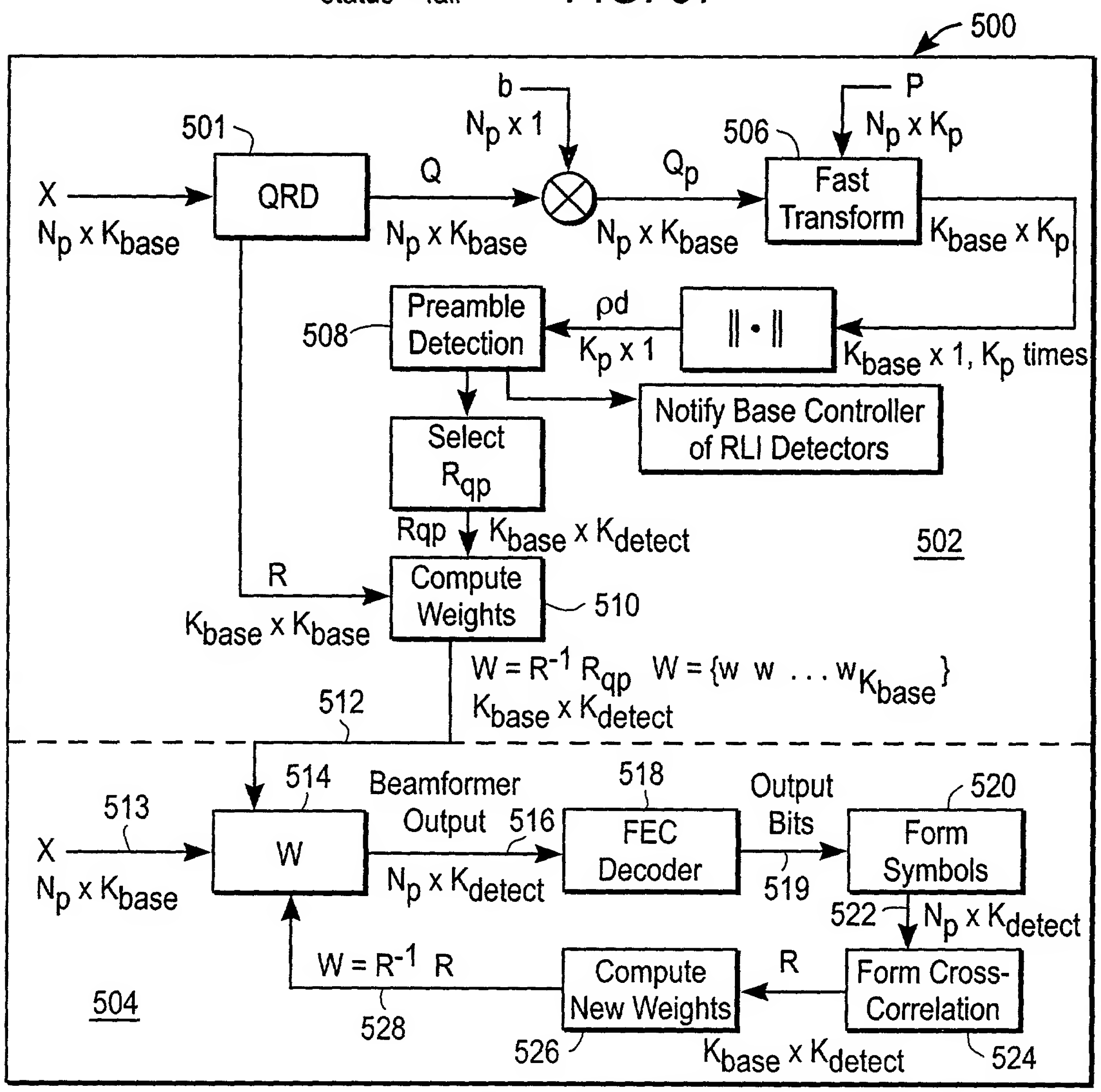

500
b
$N_p \times 1$
P
$N_p \times K_p$
501
506
X
$N_p \times K_{base}$
QRD
Q
$N_p \times K_{base}$
Q_p
$N_p \times K_{base}$
Fast Transform
$K_{base} \times K_p$
Preamble Detection
508
ρd
$K_p \times 1$
$\|\cdot\|$
$K_{base} \times 1$, K_p times
Notify Base Controller of RLI Detectors
Select R_{qp}
Rqp
$K_{base} \times K_{detect}$
502
R
Compute Weights
510
$K_{base} \times K_{base}$
$W = R^{-1} R_{qp}$ $W = \{w\ w\ \ldots w_{K_{base}}\}$
$K_{base} \times K_{detect}$
512
514
Beamformer Output
516
518
FEC Decoder
Output Bits
519
520
Form Symbols
513
X
$N_p \times K_{base}$
W
$N_p \times K_{detect}$
522
$N_p \times K_{detect}$
$W = R^{-1}$ R
528
Compute New Weights
526
R
Form Cross-Correlation
524
$K_{base} \times K_{detect}$
504

FIG. 38

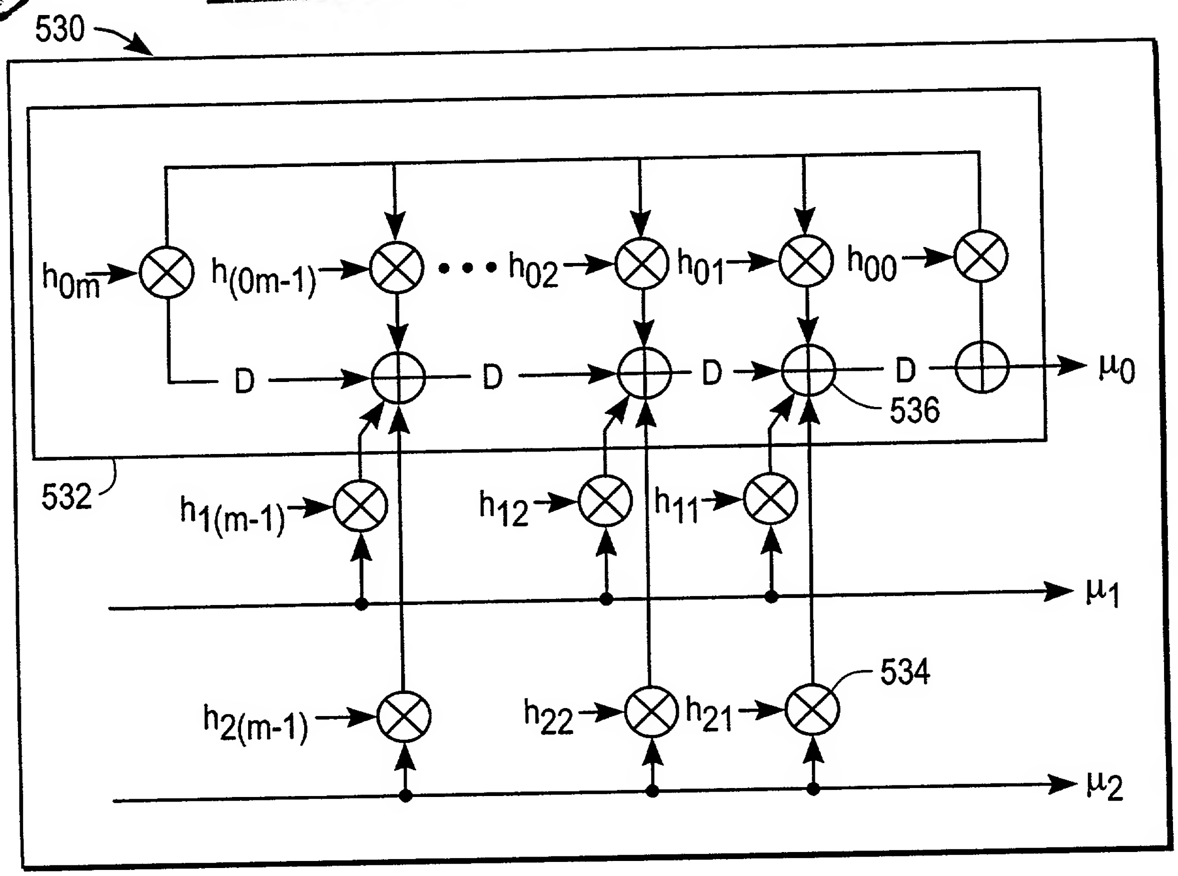

530
532
536
534
h_{0m}
$h_{(0m-1)}$
h_{02}
h_{01}
h_{00}
D
D
D
D
μ_0
$h_{1(m-1)}$
h_{12}
h_{11}
μ_1
$h_{2(m-1)}$
h_{22}
h_{21}
μ_2

FIG. 39

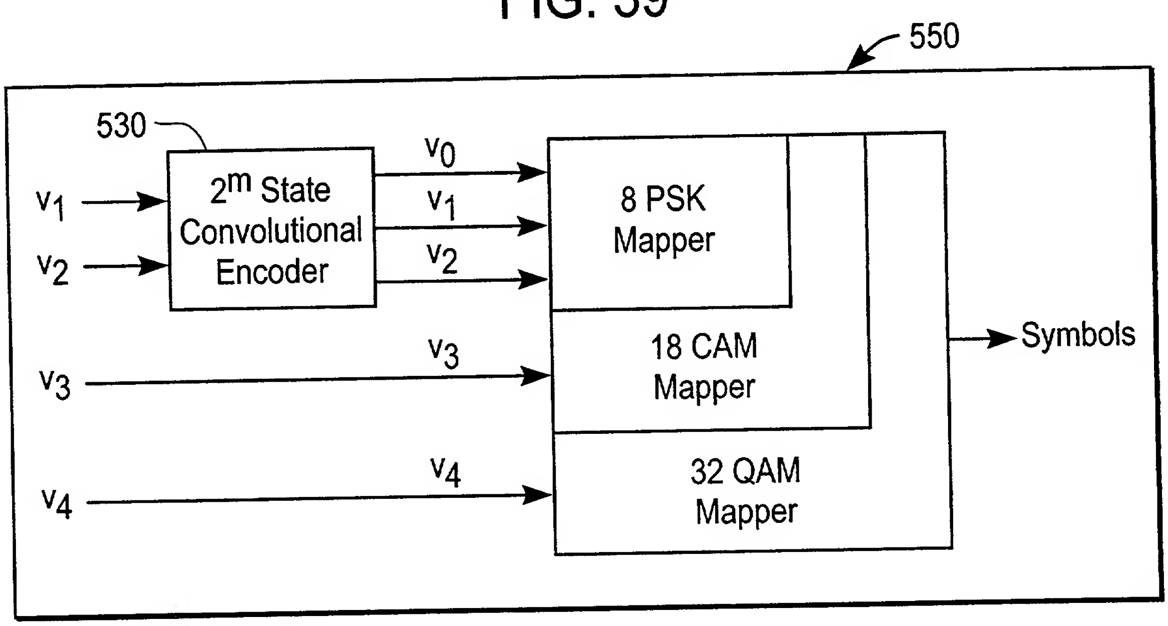

550
530
2^m State Convolutional Encoder
v_1
v_2
v_0
v_1
v_2
v_3
v_3
v_4
v_4
8 PSK Mapper
18 CAM Mapper
32 QAM Mapper
Symbols

FIG. 40

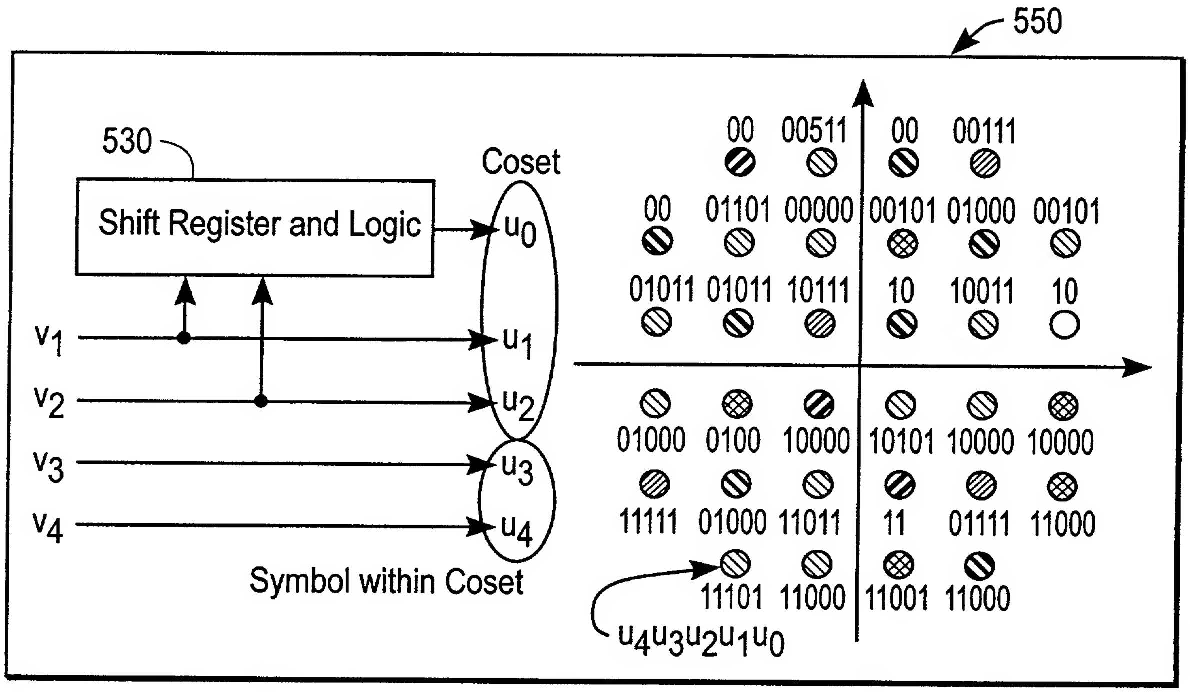

550
530
Shift Register and Logic
Coset
v1
v2
v3
v4
u0
u1
u2
u3
u4
Symbol within Coset
00 00511 00 00111
00 01101 00000 00101 01000 00101
01011 01011 10111 10 10011 10
01000 0100 10000 10101 10000 10000
11111 01000 11011 11 01111 11000
11101 11000 11001 11000
u4u3u2u1u0

FIG. 41

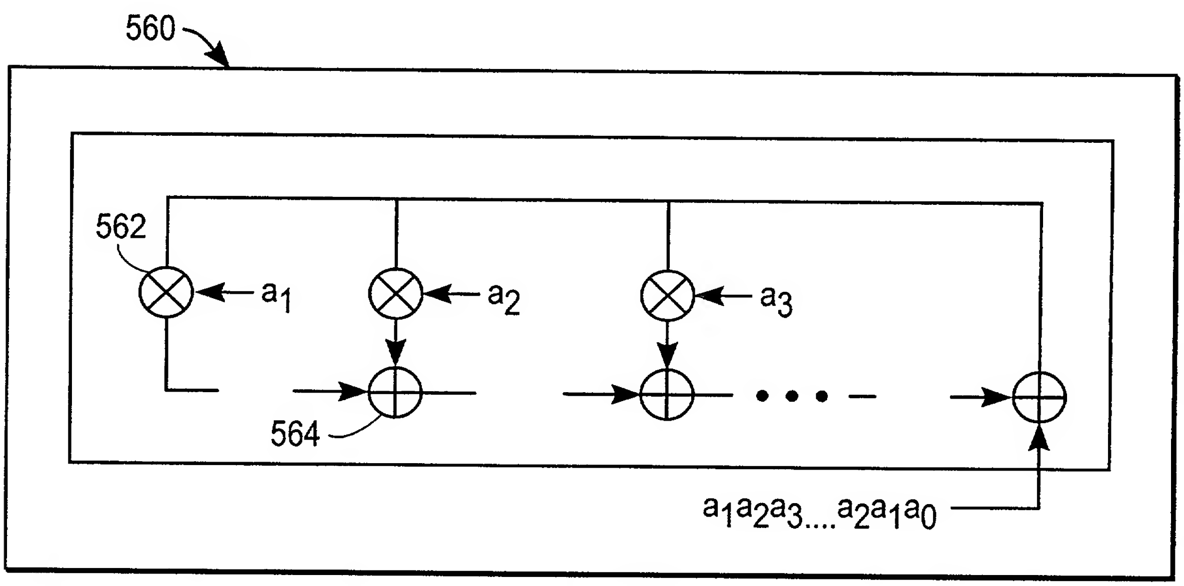

560
562
a1
a2
a3
564
a1a2a3....a2a1a0

FIG. 42

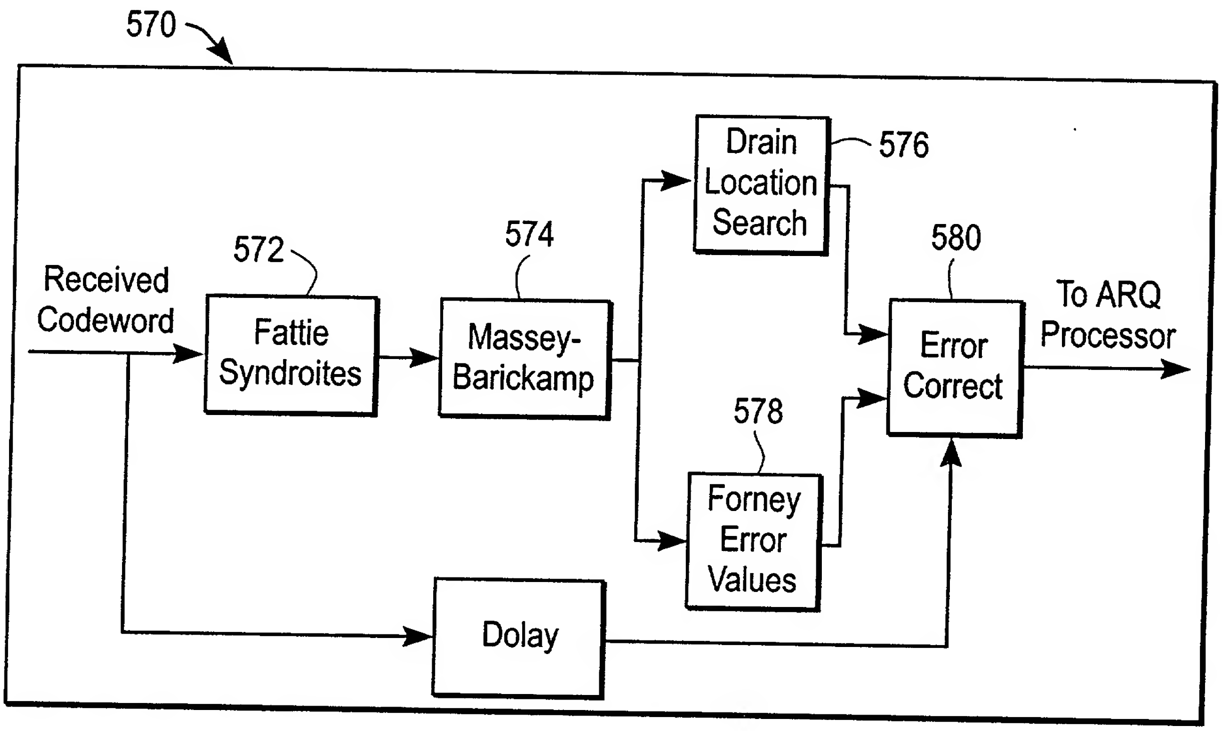

570
Received
Codeword
572
Fattie
Syndroites
574
Massey-
Barickamp
Drain
Location
Search
576
580
Error
Correct
To ARQ
Processor
578
Forney
Error
Values
Dolay

FIG. 43

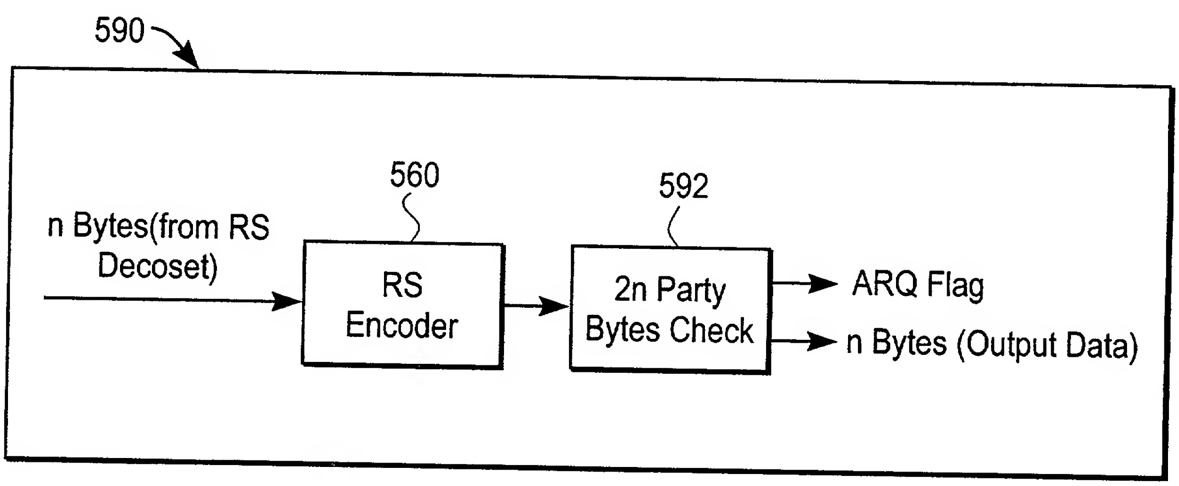

590
n Bytes(from RS
Decoset)
560
RS
Encoder
592
2n Party
Bytes Check
ARQ Flag
n Bytes (Output Data)

FIG. 44

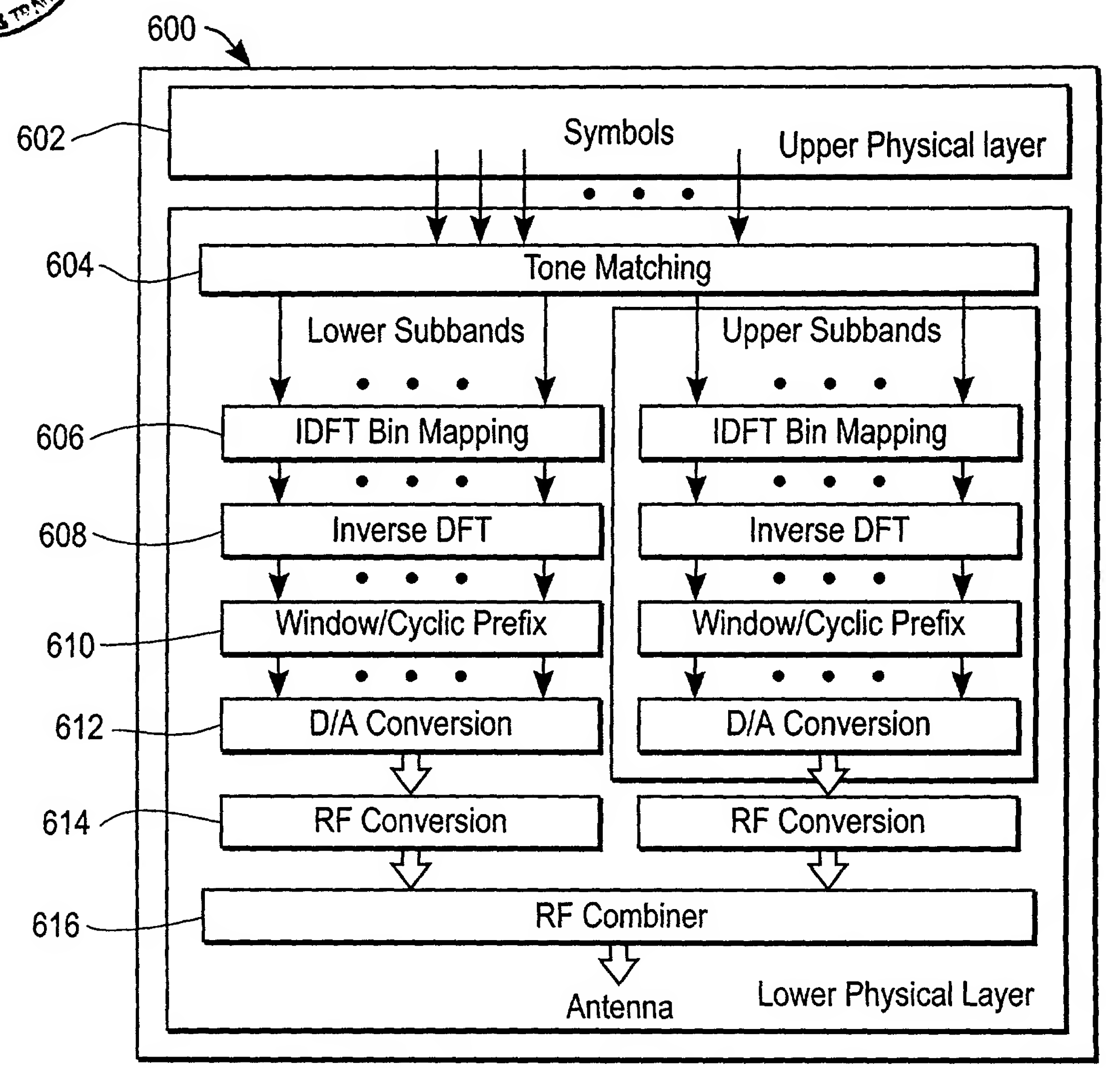

600
602
Symbols
Upper Physical layer
604
Tone Matching
Lower Subbands
Upper Subbands
606
IDFT Bin Mapping
IDFT Bin Mapping
608
Inverse DFT
Inverse DFT
610
Window/Cyclic Prefix
Window/Cyclic Prefix
612
D/A Conversion
D/A Conversion
614
RF Conversion
RF Conversion
616
RF Combiner
Antenna
Lower Physical Layer

FIG. 45

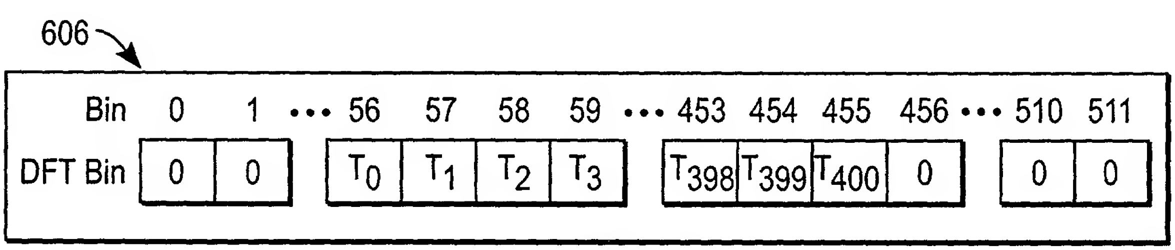

606
Bin 0 1 ••• 56 57 58 59 ••• 453 454 455 456 ••• 510 511
DFT Bin 0 0 T_0 T_1 T_2 T_3 T_{398} T_{399} T_{400} 0 0 0

FIG. 46

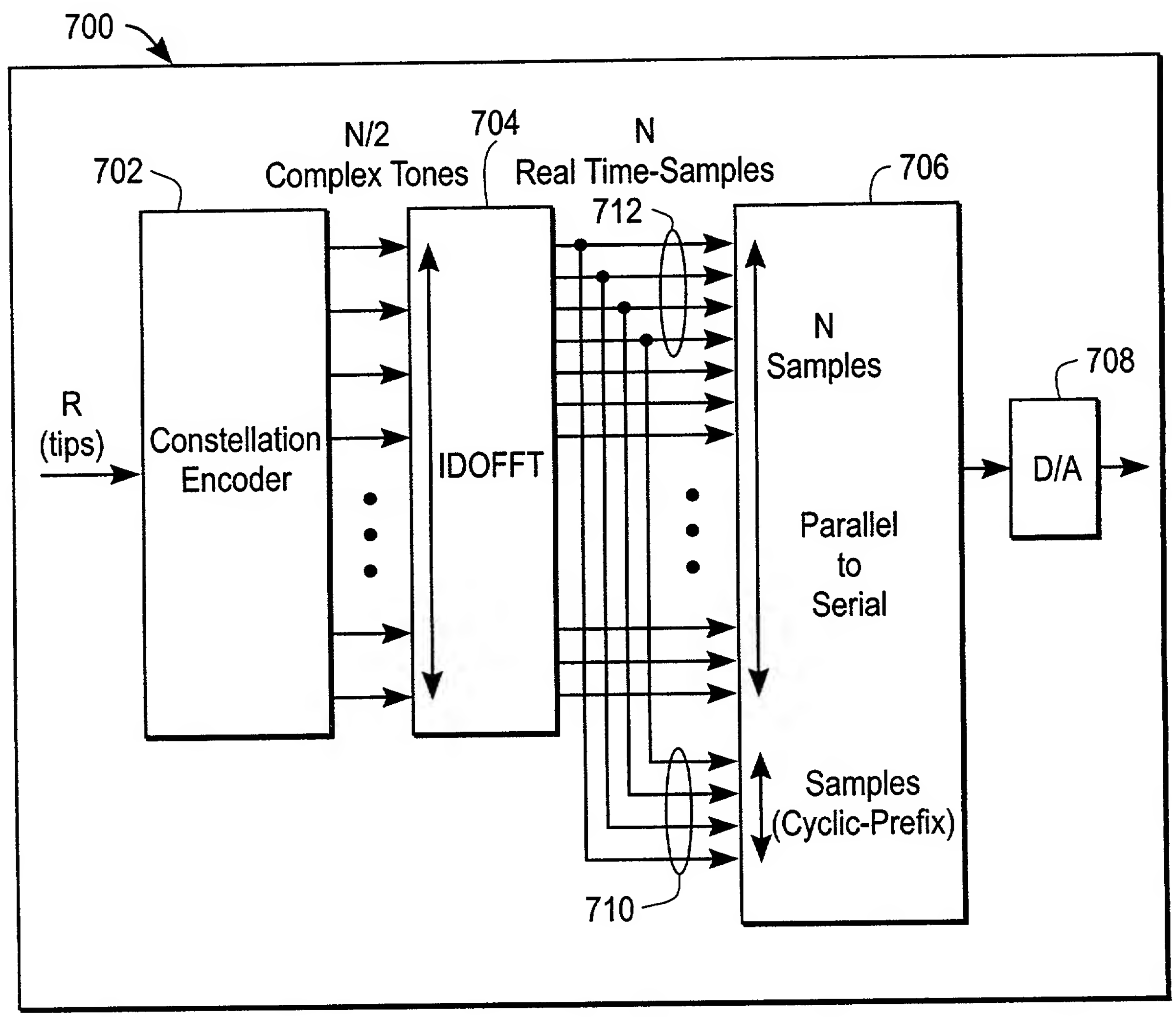

700
702
N/2
Complex Tones
704
N
Real Time-Samples
706
712
R
(tips)
Constellation
Encoder
IDOFFT
N
Samples
708
D/A
Parallel
to
Serial
Samples
(Cyclic-Prefix)
710

FIG. 47

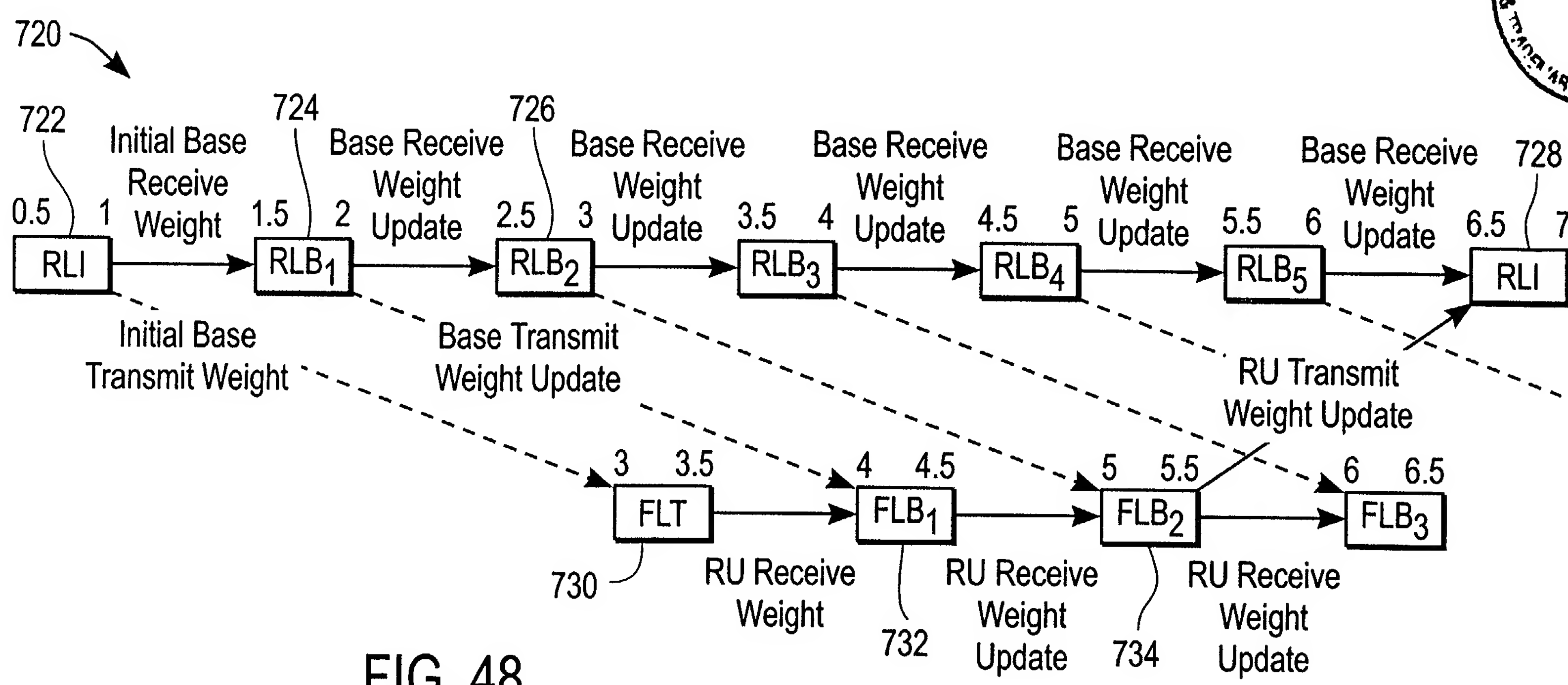

720
722
0.5
RLI
1
Initial Base Receive Weight
724
1.5
RLB1
2
Base Receive Weight Update
726
2.5
RLB2
3
Base Receive Weight Update
3.5
RLB3
4
Base Receive Weight Update
4.5
RLB4
5
Base Receive Weight Update
5.5
RLB5
6
Base Receive Weight Update
728
6.5
RLI
7
Initial Base Transmit Weight
Base Transmit Weight Update
RU Transmit Weight Update
3
FLT
3.5
730
RU Receive Weight
4
FLB1
4.5
732
RU Receive Weight Update
5
FLB2
5.5
734
RU Receive Weight Update
6
FLB3
6.5

FIG. 48

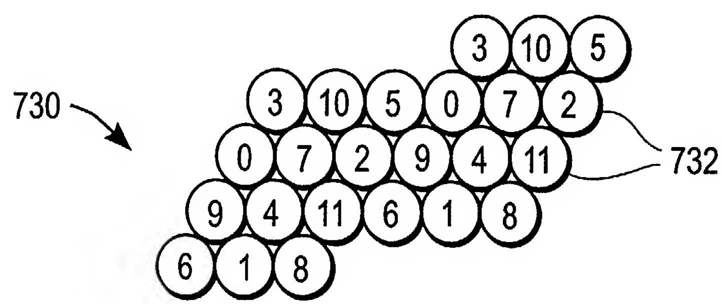

730
3 10 5
3 10 5 0 7 2
0 7 2 9 4 11
9 4 11 6 1 8
6 1 8
732

FIG. 49

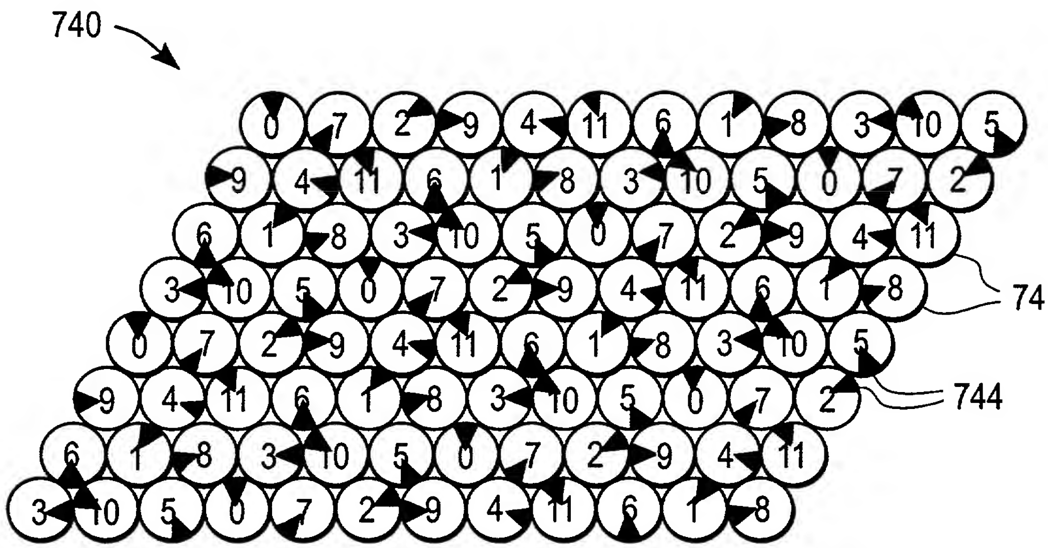


FIG. 50

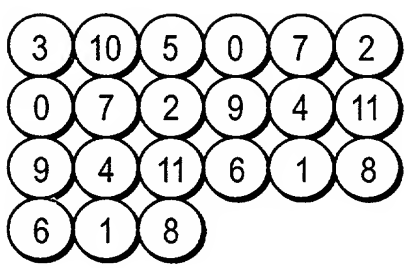


FIG. 51

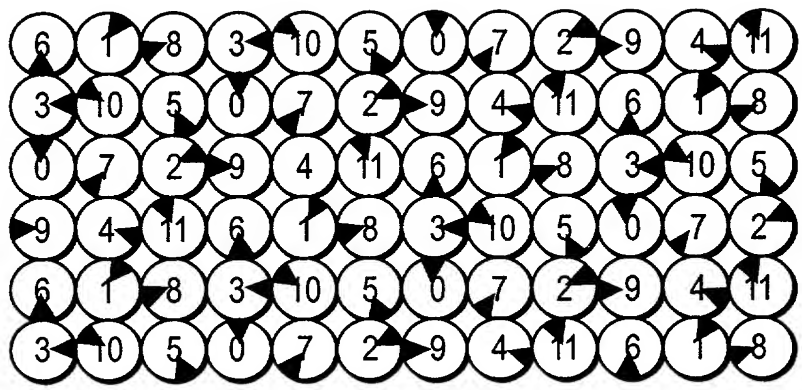

6 1 8 3 10 5 0 7 2 9 4 11
3 10 5 0 7 2 9 4 11 6 1 8
0 7 2 9 4 11 6 1 8 3 10 5
9 4 11 6 1 8 3 10 5 0 7 2
6 1 8 3 10 5 0 7 2 9 4 11
3 10 5 0 7 2 9 4 11 6 1 8

FIG. 52

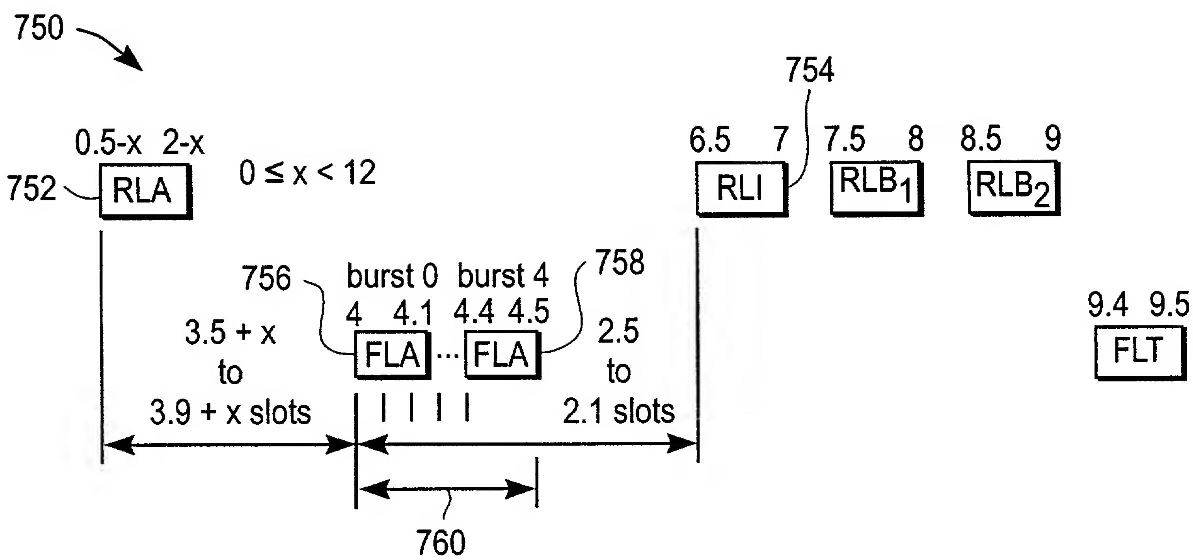

750
752
0.5-x 2-x
RLA
$0 \le x < 12$
754
6.5 7
RLI
7.5 8
RLB_1
8.5 9
RLB_2
756
burst 0 burst 4
4 4.1 4.4 4.5
758
FLA
FLA
3.5 + x
to
3.9 + x slots
2.5
to
2.1 slots
9.4 9.5
FLT
760

FIG. 53

Band	Bandwidth (MHz)	Subbands	Guard Bands (MHz)
WCS (A & B)	2 x 5	3	0.625
WCS (C/D)	2 x 5	2	1.250
MMDS	2 x 12	8	1.000

FIG. 54

Band of Interest	Channel Bandwidth (MHz)	Number of Subbands	Active Bandwidth (MHz)	Guard Band on each side of active band (MHz)
UHF, WCS PCS	5	3	3.75	0.625
	10	7	8.75	0.625
	15	10	12.5	1.25
MMDS	3	2	2.5	0.25
	6	4	5	0.5
	12	8	10	1
3.5 GHz	3.5	2	2.5	0.5
	7	4	5	1
	14	8	10	2
3.65 GHz	25	16	20	2.5

FIG. 55

Tone per burst	16	16	16	16	16	16
Information bits per tone	4	4	3	3	2	2
Bits per burst	64	64	48	48	32	32
Bursts per slot	5	4	5	4	5	4
Bits per bearer slot	320	256	240	192	160	128
Bits per frame	1600	1280	1200	960	800	640
Partition rate (kbps)	80	64	60	48	40	32
Full rate (kbps)	1920	1536	1440	1152	960	768

FIG. 56

i_1 \ i_0	i_0=0	i_0=1	i_0=2	i_0=3	i_0=4	i_0=5	i_0=6	i_0=7	...	i_0=62	i_0=63
i_1=0	X	a=0	a=64	a=128	a=192	...	a=3712	a=3776	a=3840	a=3904	a=3968
i_1=1	a=3969	X	a=1	a=65	a=129	...	a=3649	a=3713	a=3777	a=3841	a=3905
i_1=2	a=3906	a=3970	X	a=2	a=66	...	a=3586	a=3650	a=3714	a=3778	a=3842
i_1=3	a=3843	a=3907	a=3971	X	a=3	...	a=3523	a=3587	a=3651	a=3715	a=3779
i_1=4	a=3780	a=3844	a=3908	a=3972	X	...	a=3460	a=3524	a=3588	a=3652	a=3716
i_1=5	...	...	...	...	...	...	...	...	...	...	...
i_1=6	a=315	a=379	a=443	a=507	a=571	...	X	a=59	a=123	a=187	a=251
i_1=7	a=252	a=316	a=380	a=444	a=508	...	a=4028	X	a=60	a=124	a=188
...	a=189	a=253	a=317	a=381	a=445	...	a=3965	a=4029	X	a=61	a=125
i_1=62	a=126	a=190	a=254	a=318	a=382	...	a=3902	a=3966	a=4030	X	a=62
i_1=63	a=63	a=127	a=191	a=255	a=319	...	a=3839	a=3903	a=3967	a=4031	X

FIG. 57

```
function fli = make_fli (codeword_descriptor)
% function fli = make_fli (codeword_descriptor)
% Synthesize a scaled 16 by 1 FLI codeword.
% 0 <= codeword_descriptor < 4096

% select the octal digits from the codeword descriptor
i0 = bitand (codeword_descriptor, -7);
i1 = bitand (bitshift (codewortd_descriptor, -3), 7);
i2 = bitand (bitshift (codeword_descriptor, -6), 7);
i3 = bitand (bitshift (codeword_descriptor, -9), 7);
generatingVector = [i0, i1, i2, i3] % generating vector

% the following kronecker basis function provides 4096 total codes
% and is based on an 8-star constellation
h = [ ...
        1.1923+0.2372j, 2.0960+0.4169j, 1.1923+0.2372j,  2.0960+0.4169j, ...
        1.1923+0.2372j, 2.0960+0.4169j, 1.1923+0.2372j,  2.0960+0.4169j; ...
        2.0960+0.4169j, 0.6754+1.0108j,-0.4169+2.0960j,-1.0108+0.6754j, ...
       -2.0960+0.4169j,-0.6754-1.0108j,  0.4169-2.0960j,  1.0108 -0.6754j;

% make the kronecker codeword
fli = 1;
for jj=1:4
        fli = kron (h(:, generatingVector(jj)+1), % matlab is one based
end

% quantize the codeword
fli = round (fli);
```

FIG. 58

700

762

```
% fls_super_results_12.m
% Lower 12 bits are the base tones, upper 4 bits are the superframe tones.
% First index (row) is the base, second (column) is the superframe

Codeword = [ ...
 23125 39509 27221 55893  6741 43605 47701 10837 51797 31317 59989 19029; ...
 40269 36173 64845 44365 56653 11597 48461 15693 27981 60749 52557 32077; ...
 47781 60069 27301 15013 10917 39589 51877  2725 35493 19109 43685 55973; ...
 13669 54629  5477 34149 62821 21861  9573 38245 42341 46437 30053 50533; ...
 27309 10925 55981 43693 47789 51885  6829 35501 15021 19117 39597 23213; ...
 21813 38197 34101  5429 42293 54581  9525 62773 46389 17717 50485 58677; ...
 27477 56149 11093 43861 19285 39765  6997 23381 52053 35669 60245 47957; ...
 42389 17813 46485 50581 21909  1429  9621 62869 30101 45677 26005 58773; ...
 42709 38613 46805 14037 18133 50901  5845 22229 54997 59093 34517 30421; ...
 38217 46409 25929 42313  5449  9545 50505 13641 54601 17737 21833 30025; ...
  4693 12885 21077 16981 53845 41557 49749 62037 45653 29269 25173 37461; ...
 59049 34473  5801  9897 54953 13993 26281 18089 38569 42665 46761 50857; ...
];

%  5A55  9A55  6A55  DA55  1A55  AA55  BA55  2A55  CA55  7A55  EA55  4A55
%  9D4D  8D4D  FD4D  AD4D  DD4D  2D4D  BD4D  3D4D  6D4D  ED4D  CD4D  7D4D
%  BAA5  EAA5  6AA5  3AA5  2AA5  9AA5  CAA5  0AA5  8AA5  4AA5  AAA5  DAA5
%  3565  D565  1565  8565  F565  5565  2565  9565  A565  B565  7565  C565
%  6AAD  2AAD  DAAD  AAAD  BAAD  CAAD  1AAD  8AAD  3AAD  4AAD  9AAD  5AAD
%  5535  9535  8535  1535  A535  D535  2535  F535  B535  4535  C535  E535
%  6B55  DB55  2B55  AB55  4B55  9B55  1B55  5B55  CB55  8B55  EB55  BB55
%  A595  4595  B595  C595  5595  0595  2595  F595  7595  D595  6595  E595
%  A6D5  96D5  B6D5  36D5  46D5  C6D5  16D5  56D5  D6D5  E6D5  86D5  76D5
%  9549  B549  6549  A549  1549  2549  C549  3549  D549  4549  5549  7549
%  1255  3255  5255  4255  D255  A255  C255  F255  B255  7255  6255  9255
%  E6A9  86A9  16A9  26A9  D6A9  36A9  66A9  46A9  96A9  A6A9  B6A9  C6A9

Nb = 12; % Number of tones in base
Ns =  4; % Number of tones in superframe sequence
Nt = 16; % Total number of tones
```

764

FIG. 59

```
function fls = make_fls (base, superframe)
% function fls = make_fls (base, superframe)
% base is the base offset code and varies from 0 to 11
% superframe is the slot sequence number and varies from 0 to 11

fls_super_results_12 % read in the codeword descriptor array

t = zeros (Nt, 1);
for jj=1:Nt
        t(jj) = 2^(jj-1); % form a vector of walking ones
end

cw = codeword (base+1, superframe+1); % select codeword descriptor
bv = (bitand(cw,t) ~= 0) * 2 - 1; % make BPSK vector
fls = (15 + 15j) * bv; % scale the BPSK vector
```

FIG. 60

770

772a, 772b, 774

Partition			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Frame	Slot	Burst																									
0	0	0 3	0 1						0 1						0 1						0 1						
0	1	0 3		1 0						1 0						1 0						1 0					
0	2	0 3			2 3						2 3						2 3						2 3				
0	3	0 3				3 2						3 2						3 2						3 2			
0	4	0 3					4 5						4 5						4 5						4 5		
0	5	0 3						5 4						5 4						5 4						5 4	
1	6	0 3	6 7						6 7						6 7						6 7						
1	7	0 3		7 6						7 6						7 6						7 6					
1	8	0 3			8 9						8 9						8 9						8 9				
1	9	0 3				9 8						9 8						9 8						9 8			
1	10	0 3					10 11						10 11						10 11						10 11		
1	11	0 3						11 10						11 10						11 10						11 10	
2	12	0 3	0 1						0 1						0 1						0 1						
2	13	0 3		1 0						1 0						1 0						1 0					
2	14	0 3			2 3						2 3						2 3						2 3				
2	15	0 3				3 2						3 2						3 2						3 2			
2	16	0 3					4 5						4 5						4 5						4 5		
2	17	0 3						5 4						5 4						5 4						5 4	
3	18	0 3	6 7						6 7						6 7						6 7						
3	19	0 3		7 6						7 6						7 6						7 6					
3	20	0 3			8 9						8 9						8 9						8 9				
3	21	0 3				9 8						9 8						9 8						9 8			
3	22	0 3					10 11						10 11						10 11						10 11		
3	23	0 3						11 10						11 10						11 10						11 10	

FIG. 61

Partition			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Frame	Slot	Burst																									
0	0	0 3	0 1			1 0			0 1			1 0			0 1			1 0			0 1			1 0			
0	1	0 3																									
0	2	0 3		2 3			3 2			2 3			3 2			2 3			3 2			2 3			3 2		
0	3	0 3																									
0	4	0 3			4 5			5 4			4 5			5 4			4 5			5 4			4 5			5 4	
0	5	0 3																									
1	6	0 3	7 6			6 7			7 6			6 7			7 6			6 7			7 6			6 7			
1	7	0 3																									
1	8	0 3		9 8			8 9			9 8			8 9			9 8			8 9			9 8			8 9		
1	9	0 3																									
1	10	0 3			11 10			10 11			11 10			10 11			11 10			10 11			11 10			10 11	
1	11	0 3																									
2	12	0 3	0 1			1 0			0 1			1 0			0 1			1 0			0 1			1 0			
2	13	0 3																									
2	14	0 3		2 3			3 2			2 3			3 2			2 3			3 2			2 3			3 2		
2	15	0 3																									
2	16	0 3			4 5			5 4			4 5			5 4			4 5			5 4			4 5			5 4	
2	17	0 3																									
3	18	0 3	7 6			6 7			7 6			6 7			7 6			6 7			7 6			6 7			
3	19	0 3																									
3	20	0 3		9 8			8 9			9 8			8 9			9 8			8 9			9 8			8 9		
3	21	0 3																									
3	22	0 3			11 10			10 11			11 10			10 11			11 10			10 11			11 10			10 11	
3	23	0 3																									

FIG. 62

Partition			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Frame	Slot	Burst																									
0	0	0 3	a a						α α						Y Y						Ψ Ψ						
0	1	0 3		b b						β β						X X						Ξ Ξ					
0	2	0 3			c c						χ χ						W W						Ω Ω				
0	3	0 3				d d						δ δ						V V						ς ς			
0	4	0 3					e e						ε ε						U U						∴ ∴		
0	5	0 3						f f						ϕ ϕ						T T						| |	
1	6	0 3	S S						γ γ						g g						Σ Σ						
1	7	0 3		R R						η η						h h						[[					
1	8	0 3			Q Q						ι ι						i i						Θ Θ				
1	9	0 3				P P						φ φ						j j						Π Π			
1	10	0 3					O O						κ κ						k k						} }		
1	11	0 3						N N						λ λ						l l						]]	
2	12	0 3	L L						Λ Λ						m m						μ μ						
2	13	0 3		K K						} }						n n						ν ν					
2	14	0 3			J J						ϑ ϑ						o o						~ ~				
2	15	0 3				I I						* *						p p						π π			
2	16	0 3					H H						{ {						q q						θ θ		
2	17	0 3						G G						Γ Γ						r r						]]	
3	18	0 3	s s						Φ Φ						F F						σ σ						
3	19	0 3		t t						& &						E E						τ τ					
3	20	0 3			u u						Δ Δ						D D						υ υ				
3	21	0 3				v v						X X						c c						ϖ ϖ			
3	22	0 3					w w						⊥ ⊥						B B						ω ω		
3	23	0 3						x x						% %						A A						ξ ξ	

FIG. 63

Burst ->	0	1	2	3	4
Time slot counter modulo 6	Partition in which the RU is directed to send an RLI				
0	20	2	8	14	20
1	3	9	15	21	3
2	10	16	22	4	10
3	17	23	5	11	17
4	0	6	12	18	0
5	13	19	1	7	13

FIG. 64

Burst ->	0	1	2	3	4
Time slot counter modulo 6	Partition in which the RU is directed to send an RLI				
0	8	11	2	5	8
1	3	6	9	0	3
2	10	1	4	7	10
3	5	8	11	2	5
4	0	3	6	9	0
5	1	4	7	10	1

FIG. 65

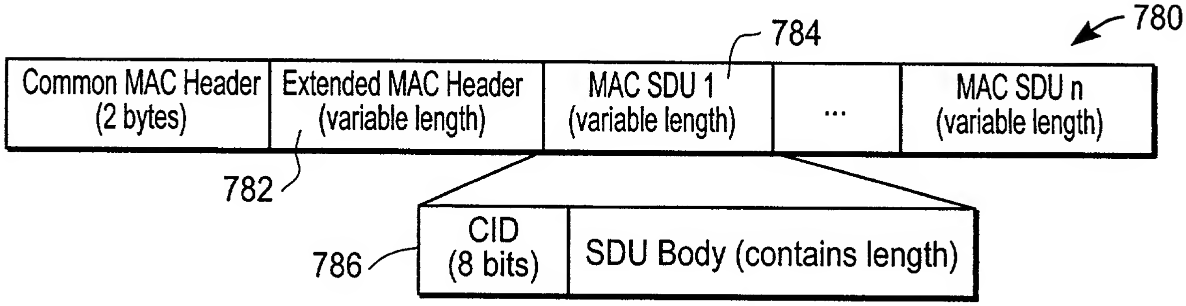


FIG. 66

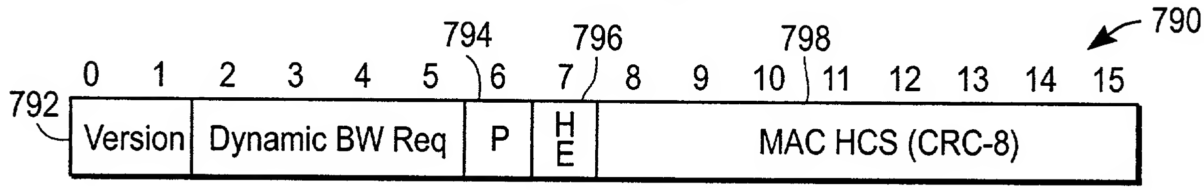


FIG. 67

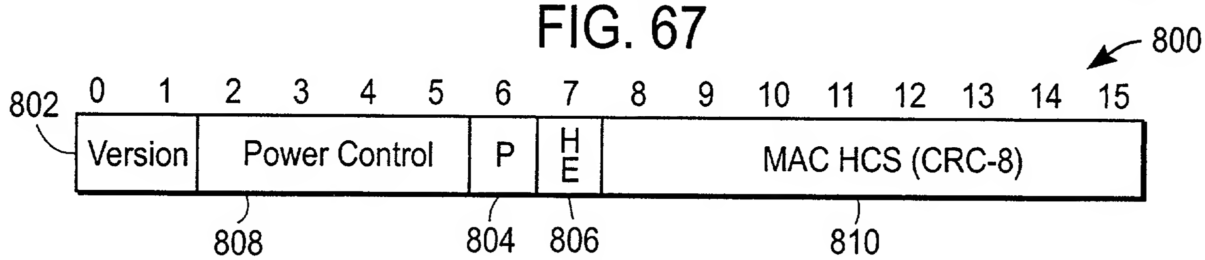


FIG. 68

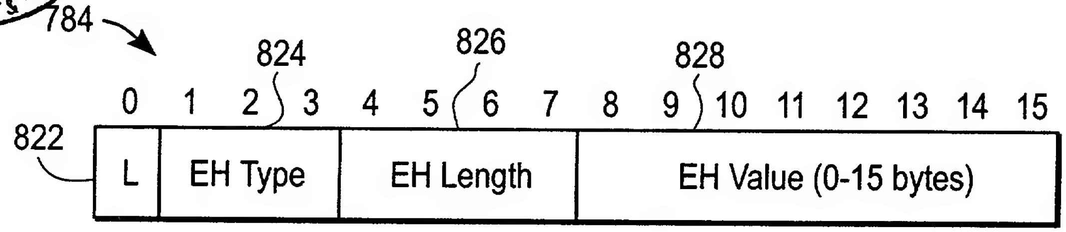


FIG. 69

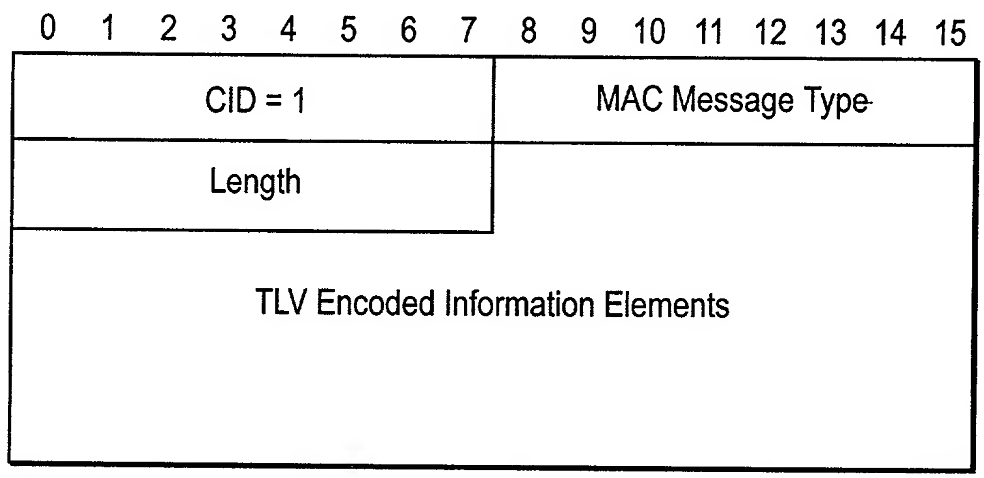
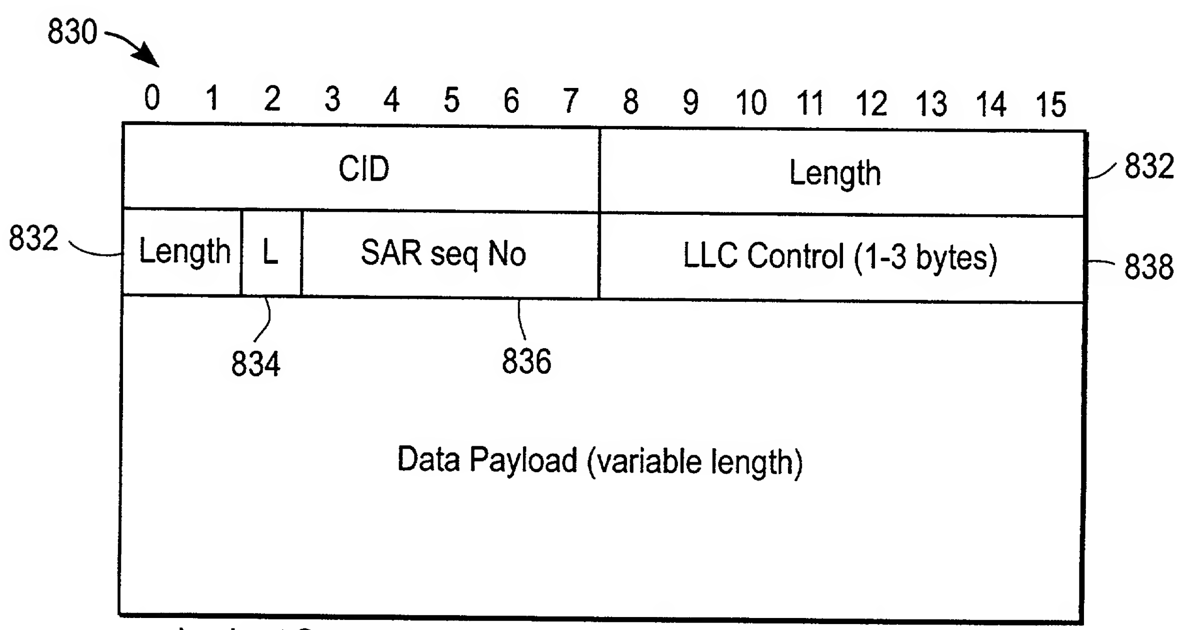


FIG. 70



- L = Last Segment

FIG. 71

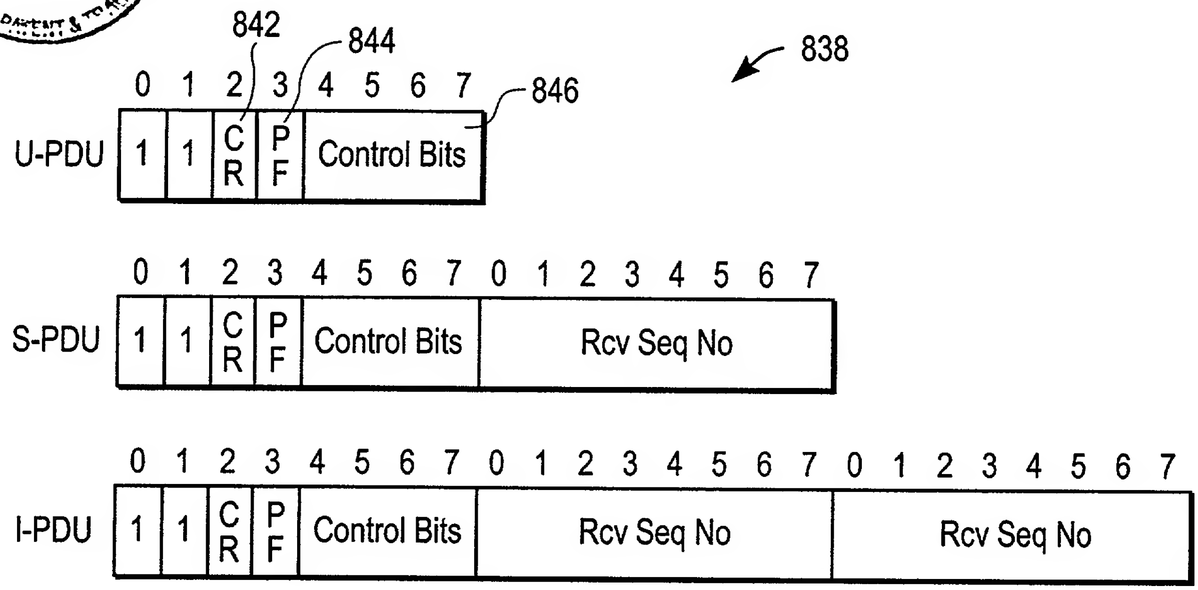


FIG. 72

Modulation Order	4 Bits/Sym		3 Bits/Sym		2 Bits/Sym	
Link Direction	Forward	Reverse	Forward	Reverse	Forward	Reverse
Bits/Symbol	4	4	3	3	2	2
Symbols/Burst	16	16	16	16	16	16
Bursts/Slot	5	5	5	4	5	4
Bits/Slot	320	256	240	192	160	128
Bytes/Slot	40	32	30	24	20	16
Slots/Frame	5	5	5	5	5	5
Bits/Frame	1600	1280	1200	960	800	640
Bytes/Frame	200	160	150	120	100	80
Viterbi Tail Byte(*)	1	1	1	1	1	1
RS Check Bytes	28	28	18	18	10	10
Common MAC Header	2	2	2	2	2	2
MAC SDU Header	169	129	129	99	87	67
Data SDU Header	6	6	6	6	6	6
Data Payload	163	123	123	93	81	61
Data Rate/Partition, kbps	65.2	49.2	49.2	37.2	32.4	24.4
Partitions/Subband	24	24	24	24	24	24
Data Rate/Subband, kbps	1564.8	1180.8	1180.8	892.8	777.6	585.6
Subband Data Rate/T1	1.02	0.77	0.77	0.58	0.51	0.38

FIG. 73

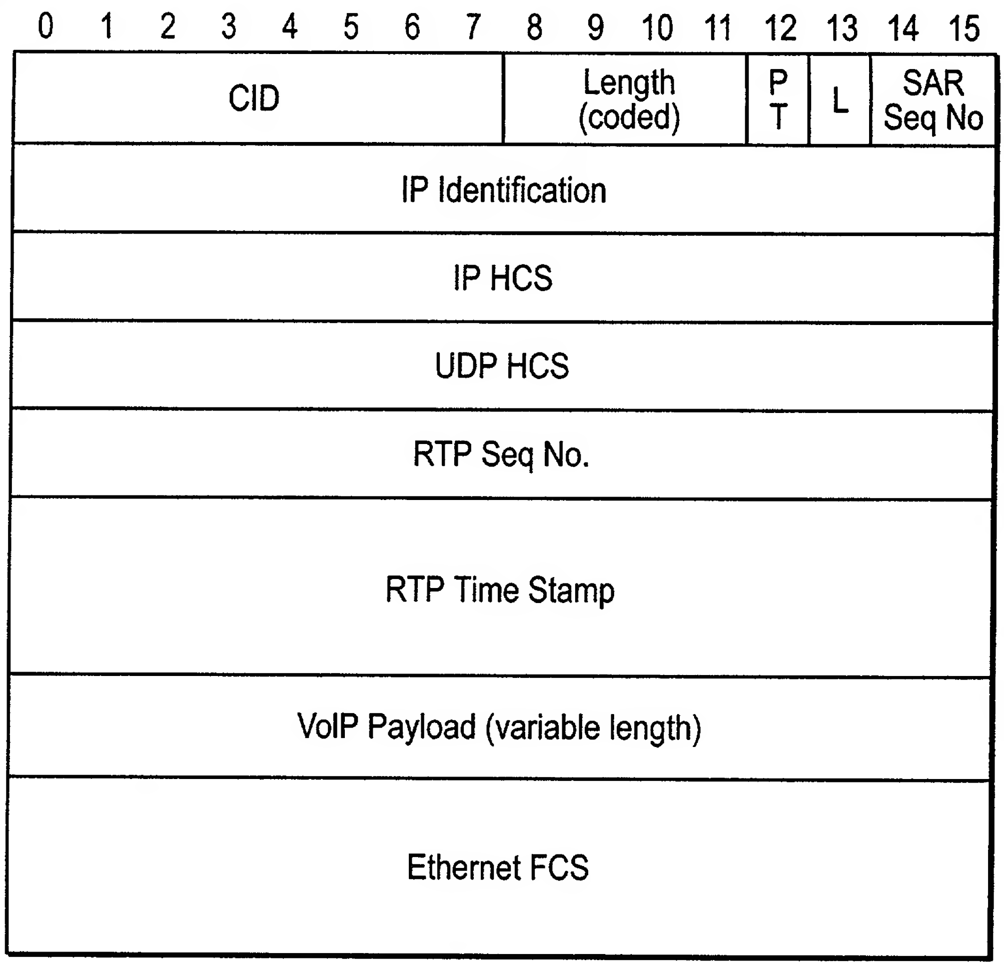

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
CID
Length (coded)
P T
L
SAR Seq No
IP Identification
IP HCS
UDP HCS
RTP Seq No.
RTP Time Stamp
VoIP Payload (variable length)
Ethernet FCS

FIG. 74

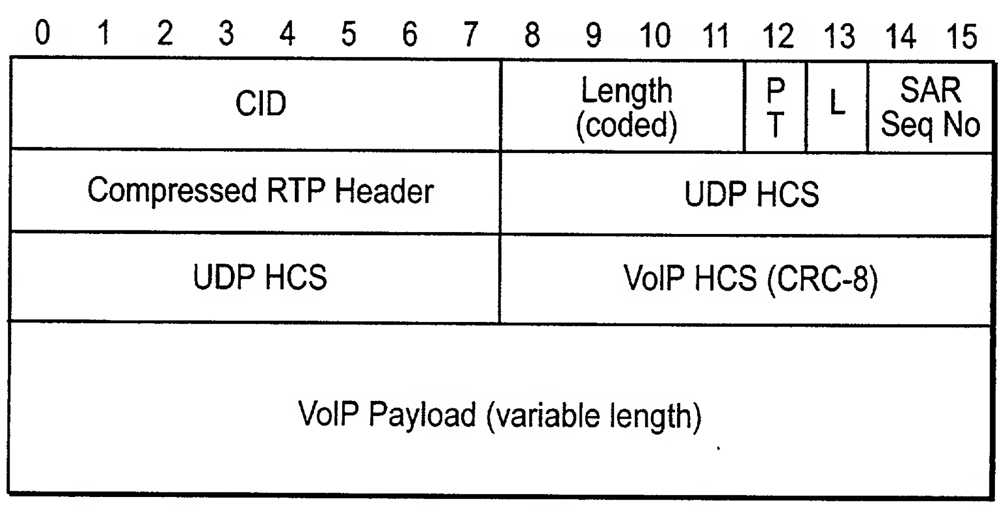

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
CID
Length (coded)
P T
L
SAR Seq No
Compressed RTP Header
UDP HCS
UDP HCS
VoIP HCS (CRC-8)
VoIP Payload (variable length)

FIG. 75

Modulation Order	4 Bits/Sym		3 Bits/Sym		2 Bits/Sym		
Link Direction	Forward	Reverse	Forward	Reverse	Forward	Reverse	
Entry Slot	–	–	–	–	–	–	10 ms
Bearer Slot 1	40	32	30	24	20	16	
Bearer Slot 2	40	32	30	24	20	16	
Common MAC Header	2	2	2	2	2	2	
Mac SDU Length	78	62	58	46	38	30	
Bearer Slot 3	40	32	30	24	20	16	10 ms
Bearer Slot 4	40	32	30	24	20	16	
Bearer Slot 5(*)	39	31	29	23	19	15	
Common MAC Header	2	2	2	2	2	2	
MAC SDU Length	117	93	87	69	57	45	

(*) Viterbi tail byte occurs in the 5th bearer slot.

FIG. 76

Frame Duration	20ms			10 ms		
Vocoder	G.711	G.726	G.729	G.711	G.726	G.729
Bit Rate, kbps	64.0	32.0	8.0	64.0	32.0	8.0
Voice Bytes	160	80	20	80	40	10
VoIP Overhead(*)	16	16	16	3	3	3
VoIP Payload Size	176	96	36	83	43	13
Voice SDU Header	2	2	2	3	3	3
4 Bits/Sym						
SDU Size Limit (RL)	129	129	129	62	62	62
No. Partitions	2	1	1/3	2	1	1/3
SDU Size	90x2	98	38	45+44	46	16
3 Bits/Sym						
SDU Size Limit (RL)	99	99	99	46	46	46
No. Partitions	2	1	1/2	2	1	1/2
SDU Size	90x2	98	38	45+44	46	16
2 Bits/Sym						
SDU Size Limit (RL)	67	67	67	30	30	30
No. Partitions	3	2	1	4	2	1
SDU Size	61x2+60	50x2	38	24x3+23	25+24	16

(*) Include RTP, UDP, IP, PPPoE, and Ethernet

FIG. 77

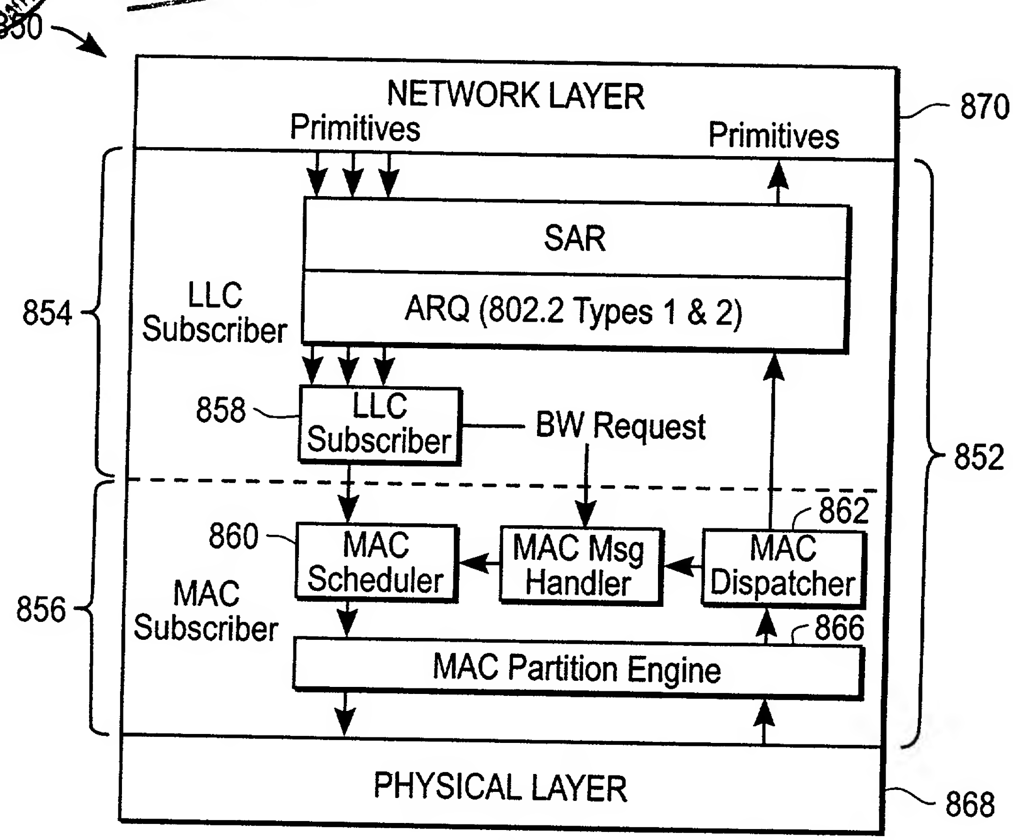

850
NETWORK LAYER
Primitives
Primitives
870
SAR
ARQ (802.2 Types 1 & 2)
854
LLC Subscriber
858
LLC Subscriber
BW Request
852
862
860
MAC Scheduler
MAC Msg Handler
MAC Dispatcher
856
MAC Subscriber
866
MAC Partition Engine
PHYSICAL LAYER
868

FIG. 78

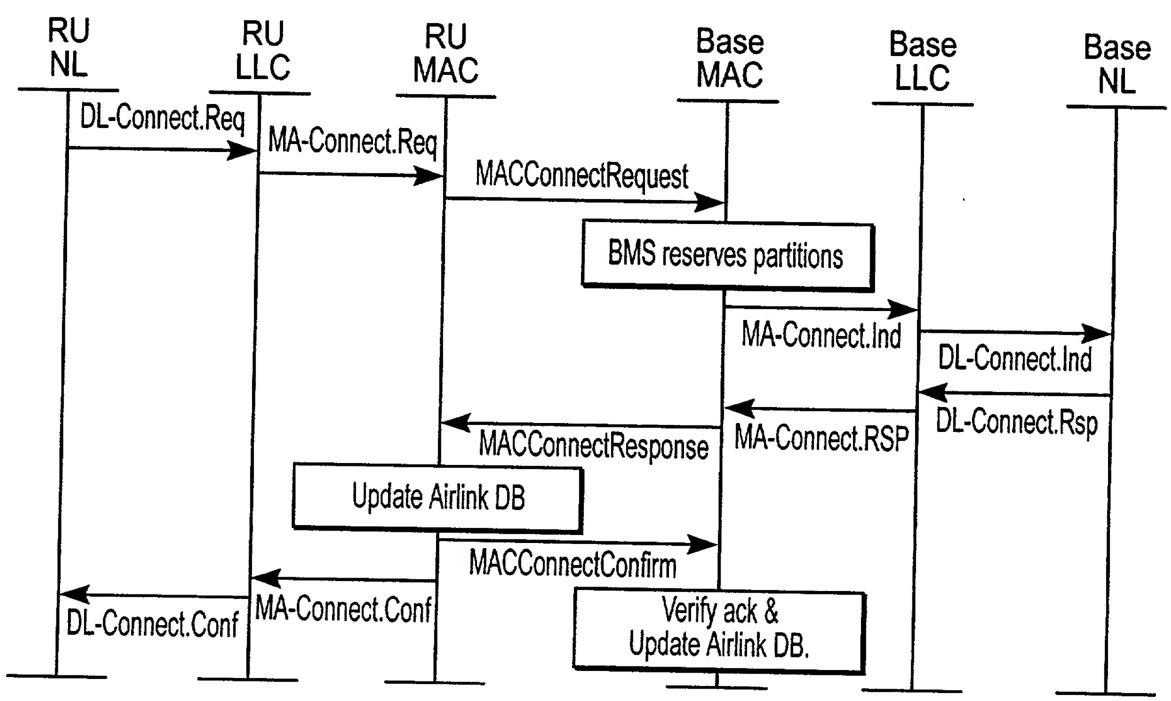

RU NL
RU LLC
RU MAC
Base MAC
Base LLC
Base NL
DL-Connect.Req
MA-Connect.Req
MACConnectRequest
BMS reserves partitions
MA-Connect.Ind
DL-Connect.Ind
DL-Connect.Rsp
MA-Connect.RSP
MACConnectResponse
Update Airlink DB
MACConnectConfirm
MA-Connect.Conf
DL-Connect.Conf
Verify ack & Update Airlink DB.

FIG. 79

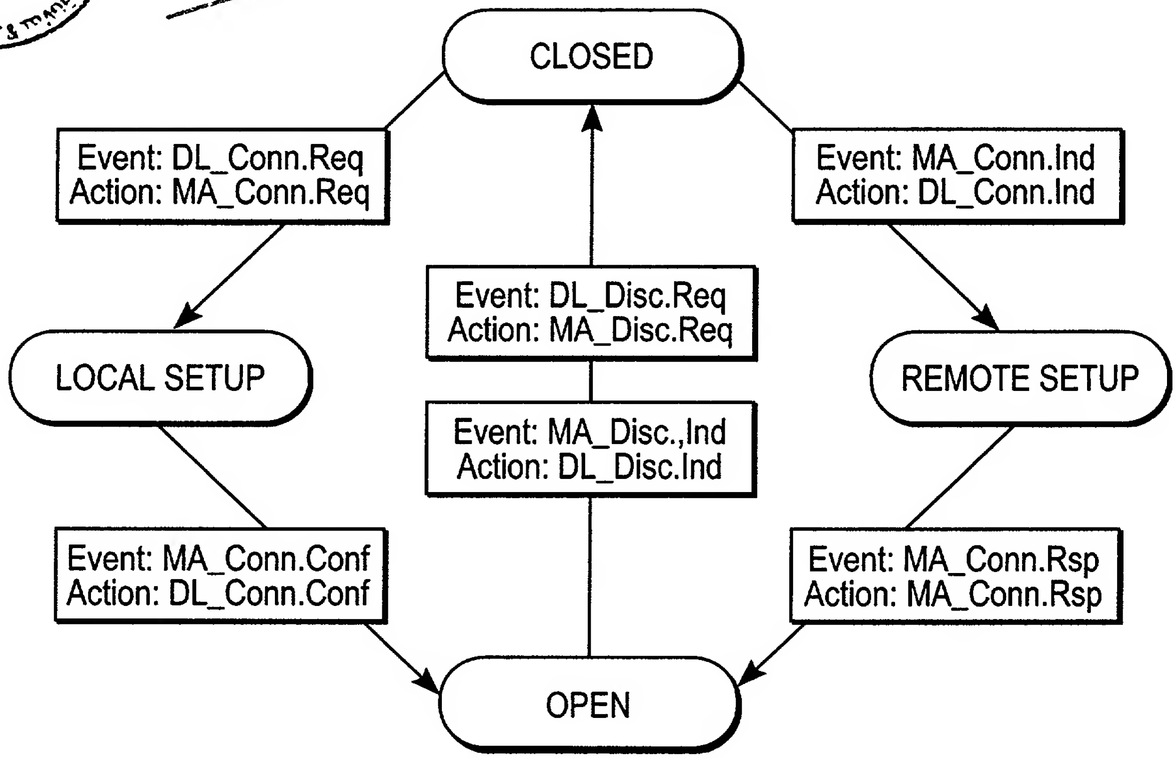

CLOSED
Event: DL_Conn.Req
Action: MA_Conn.Req
Event: MA_Conn.Ind
Action: DL_Conn.Ind
Event: DL_Disc.Req
Action: MA_Disc.Req
LOCAL SETUP
REMOTE SETUP
Event: MA_Disc.,Ind
Action: DL_Disc.Ind
Event: MA_Conn.Conf
Action: DL_Conn.Conf
Event: MA_Conn.Rsp
Action: MA_Conn.Rsp
OPEN

FIG. 80

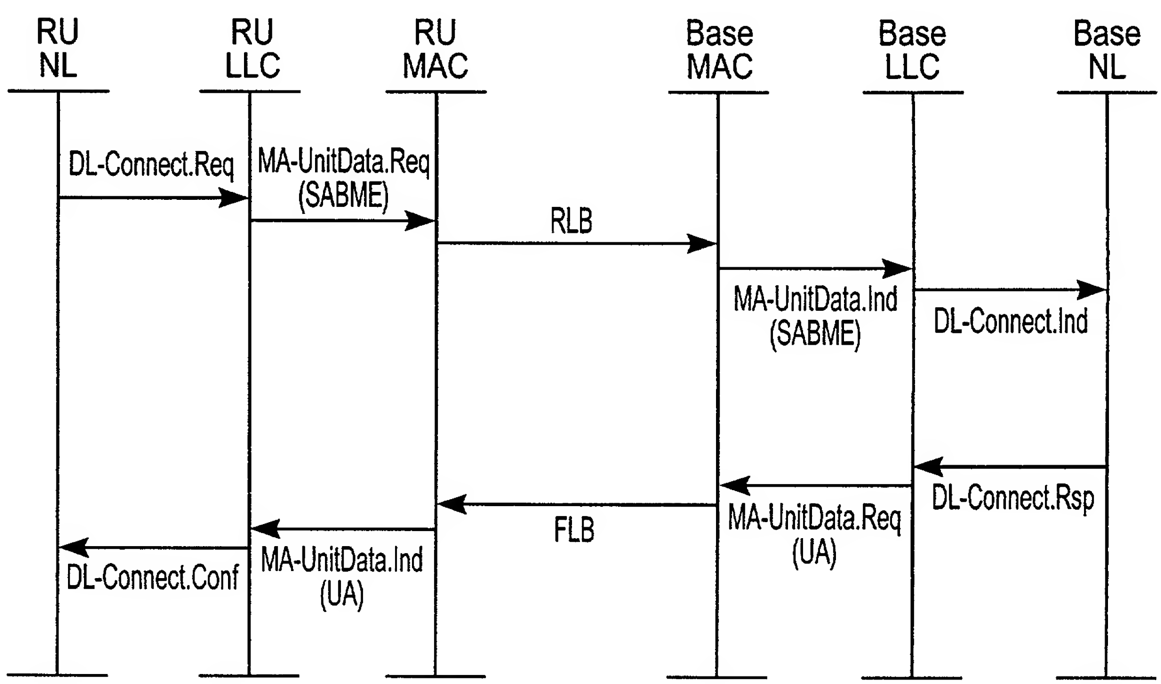

RU NL
RU LLC
RU MAC
Base MAC
Base LLC
Base NL
DL-Connect.Req
MA-UnitData.Req (SABME)
RLB
MA-UnitData.Ind (SABME)
DL-Connect.Ind
DL-Connect.Rsp
MA-UnitData.Req (UA)
FLB
MA-UnitData.Ind (UA)
DL-Connect.Conf

FIG. 81

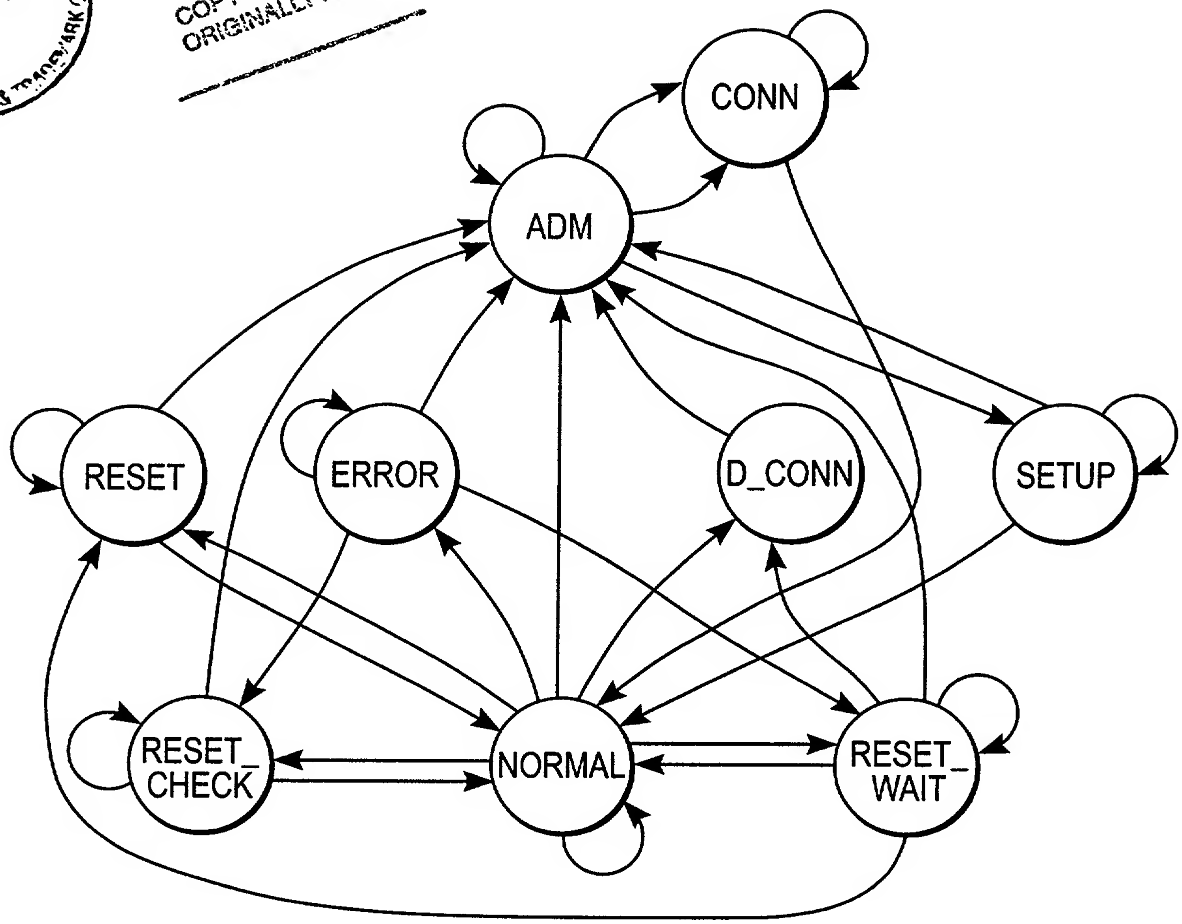


FIG. 82 (PRIOR ART)

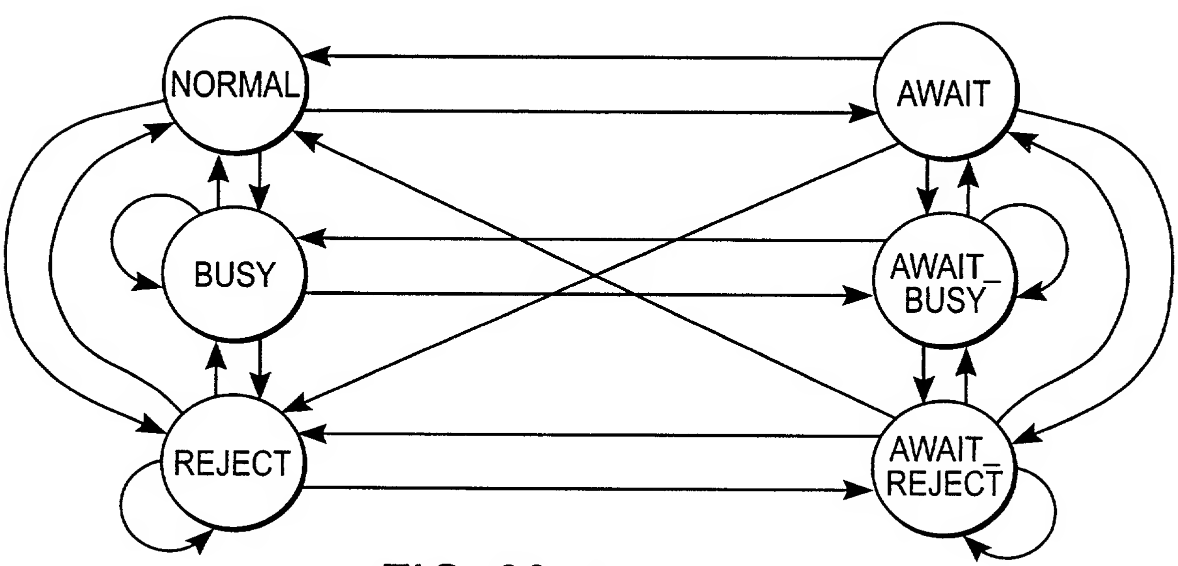


FIG. 83 (PRIOR ART)

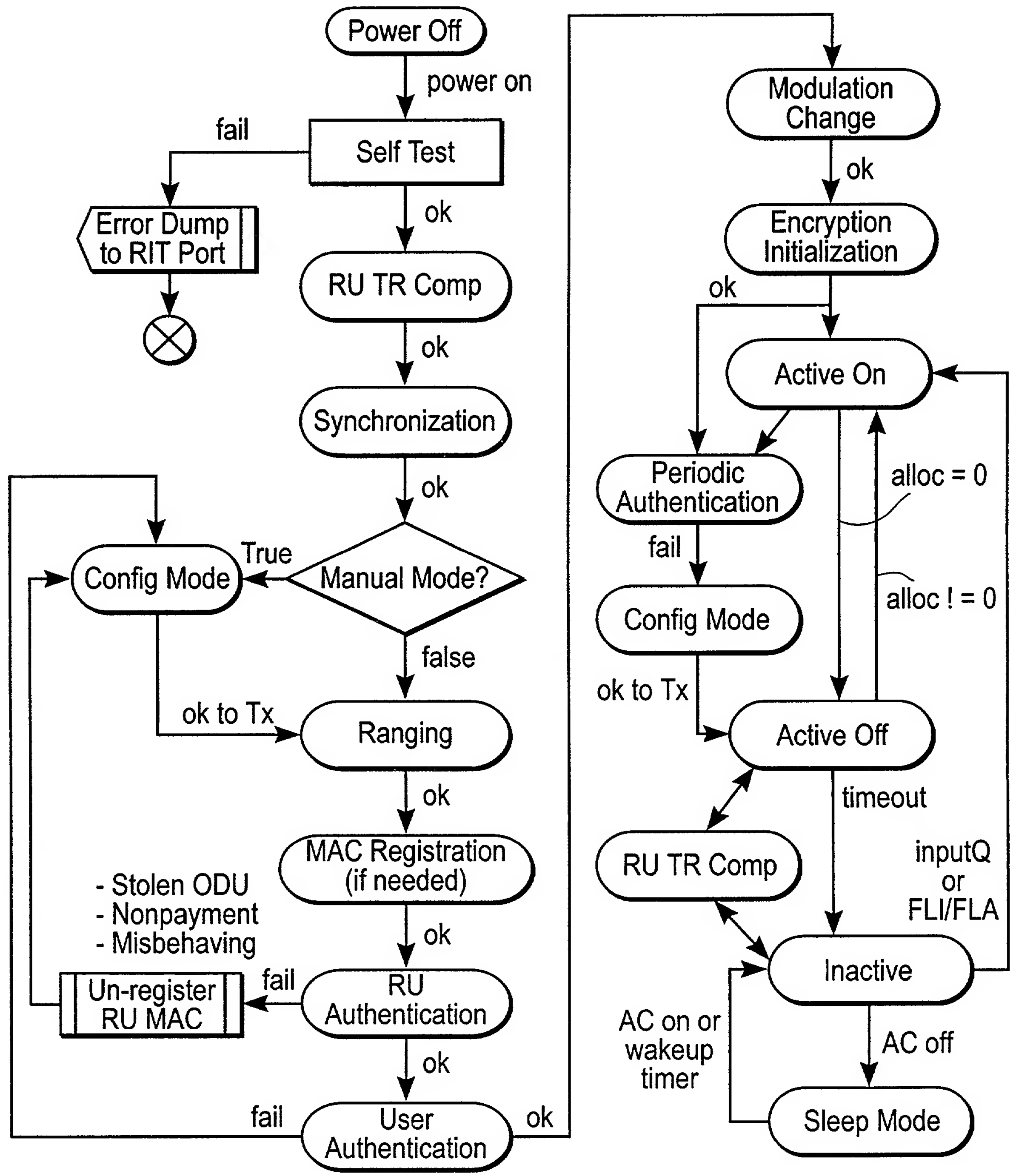

Power Off
power on
Self Test
fail
Error Dump to RIT Port
ok
RU TR Comp
ok
Synchronization
ok
Manual Mode?
True
Config Mode
false
ok to Tx
Ranging
ok
MAC Registration (if needed)
ok
- Stolen ODU
- Nonpayment
- Misbehaving
Un-register RU MAC
fail
RU Authentication
ok
fail
User Authentication
ok
Modulation Change
ok
Encryption Initialization
ok
Active On
Periodic Authentication
fail
Config Mode
ok to Tx
alloc = 0
alloc ! = 0
Active Off
timeout
RU TR Comp
inputQ or FLI/FLA
Inactive
AC on or wakeup timer
AC off
Sleep Mode

FIG. 84

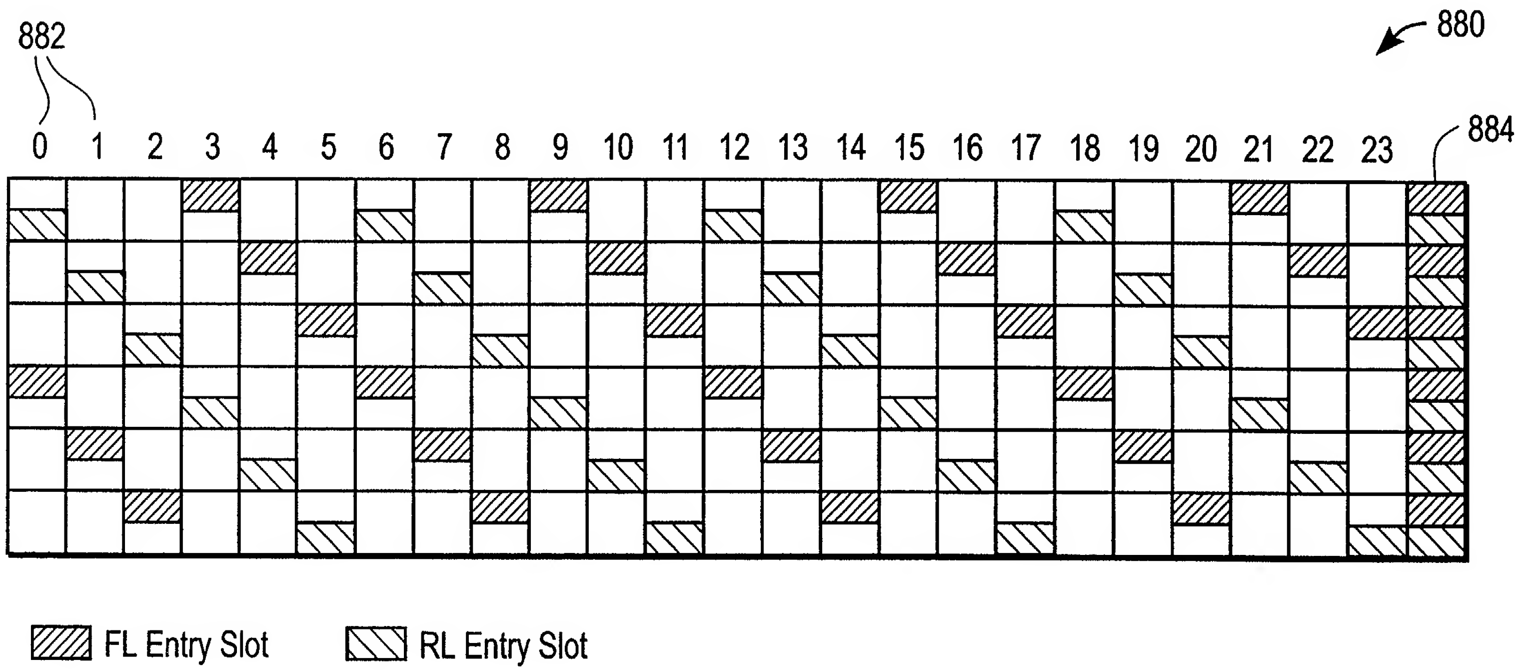

880
882
884
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
FL Entry Slot
RL Entry Slot

FIG. 85

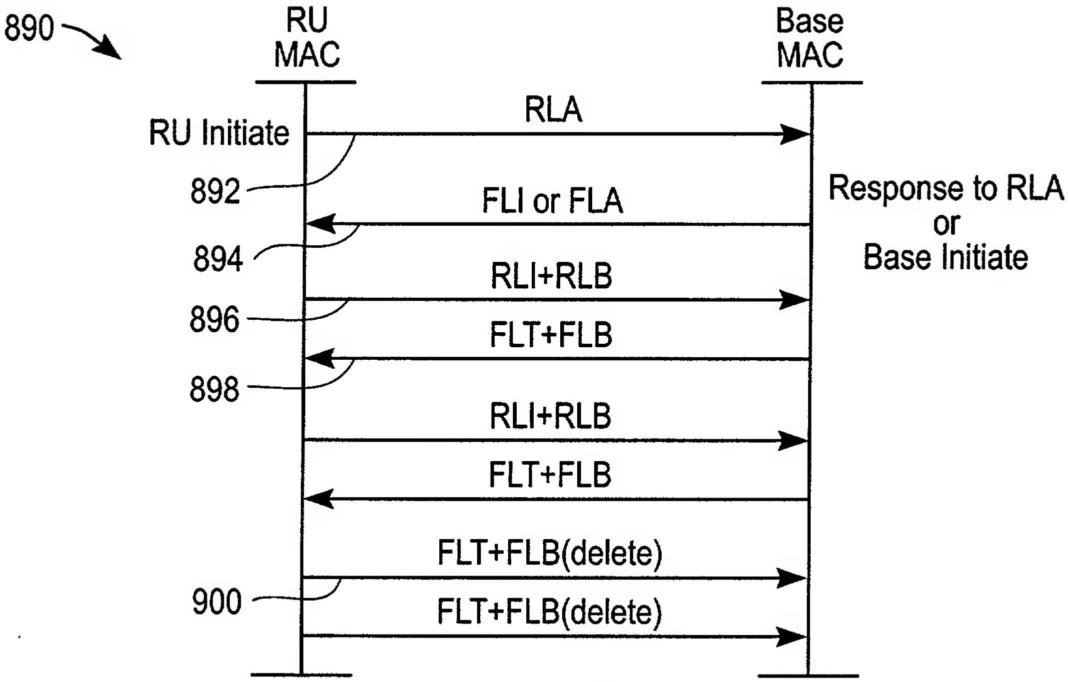

890
RU MAC
Base MAC
RU Initiate
RLA
892
FLI or FLA
Response to RLA or Base Initiate
894
RLI+RLB
896
FLT+FLB
898
RLI+RLB
FLT+FLB
FLT+FLB(delete)
900
FLT+FLB(delete)

FIG. 86

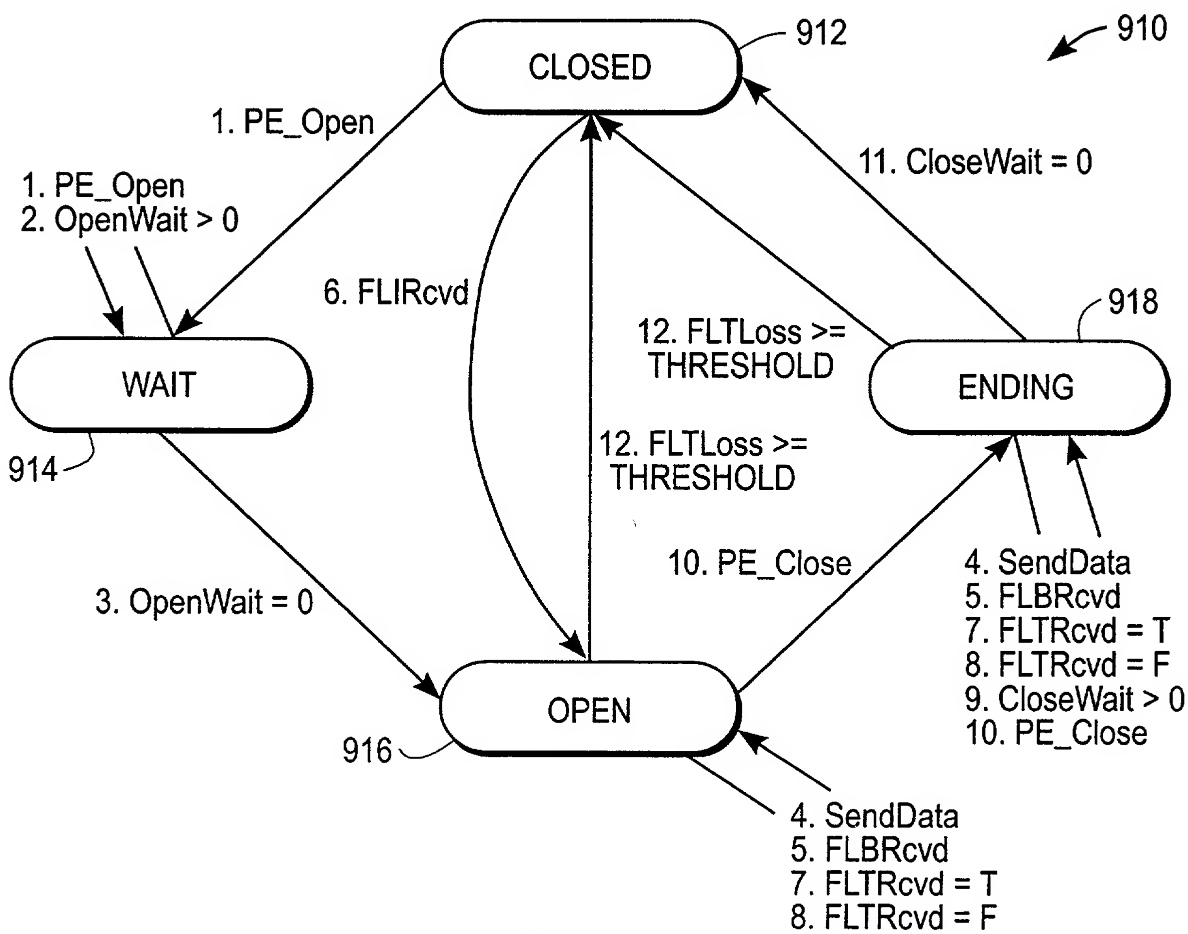

CLOSED
912
910
1. PE_Open
1. PE_Open
2. OpenWait > 0
11. CloseWait = 0
6. FLIRcvd
WAIT
914
12. FLTLoss >= THRESHOLD
12. FLTLoss >= THRESHOLD
ENDING
918
3. OpenWait = 0
10. PE_Close
4. SendData
5. FLBRcvd
7. FLTRcvd = T
8. FLTRcvd = F
9. CloseWait > 0
10. PE_Close
OPEN
916
4. SendData
5. FLBRcvd
7. FLTRcvd = T
8. FLTRcvd = F

FIG. 87

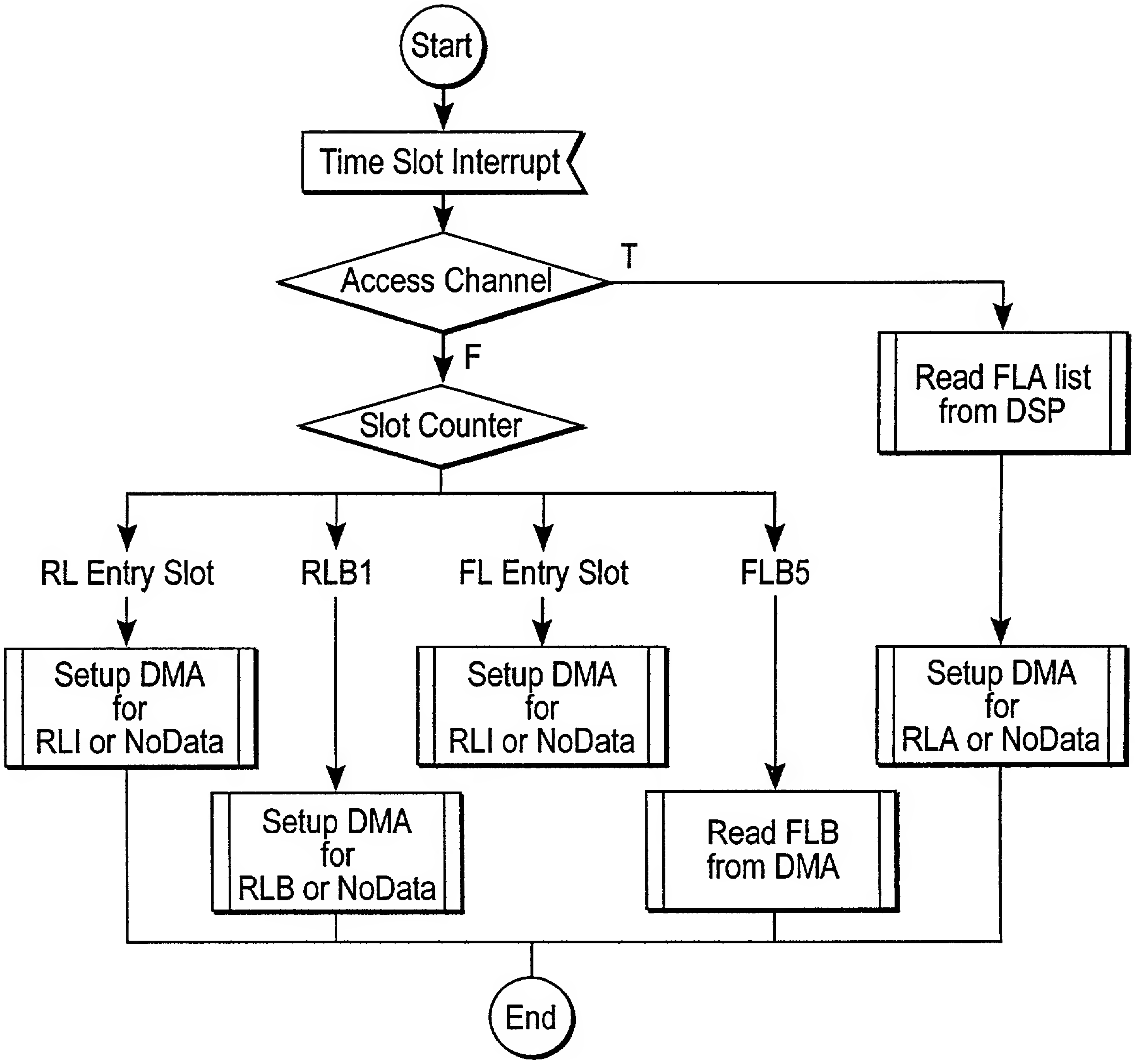

Start
Time Slot Interrupt
Access Channel
T
F
Read FLA list from DSP
Slot Counter
RL Entry Slot
RLB1
FL Entry Slot
FLB5
Setup DMA for RLI or NoData
Setup DMA for RLI or NoData
Setup DMA for RLA or NoData
Setup DMA for RLB or NoData
Read FLB from DMA
End

FIG. 88

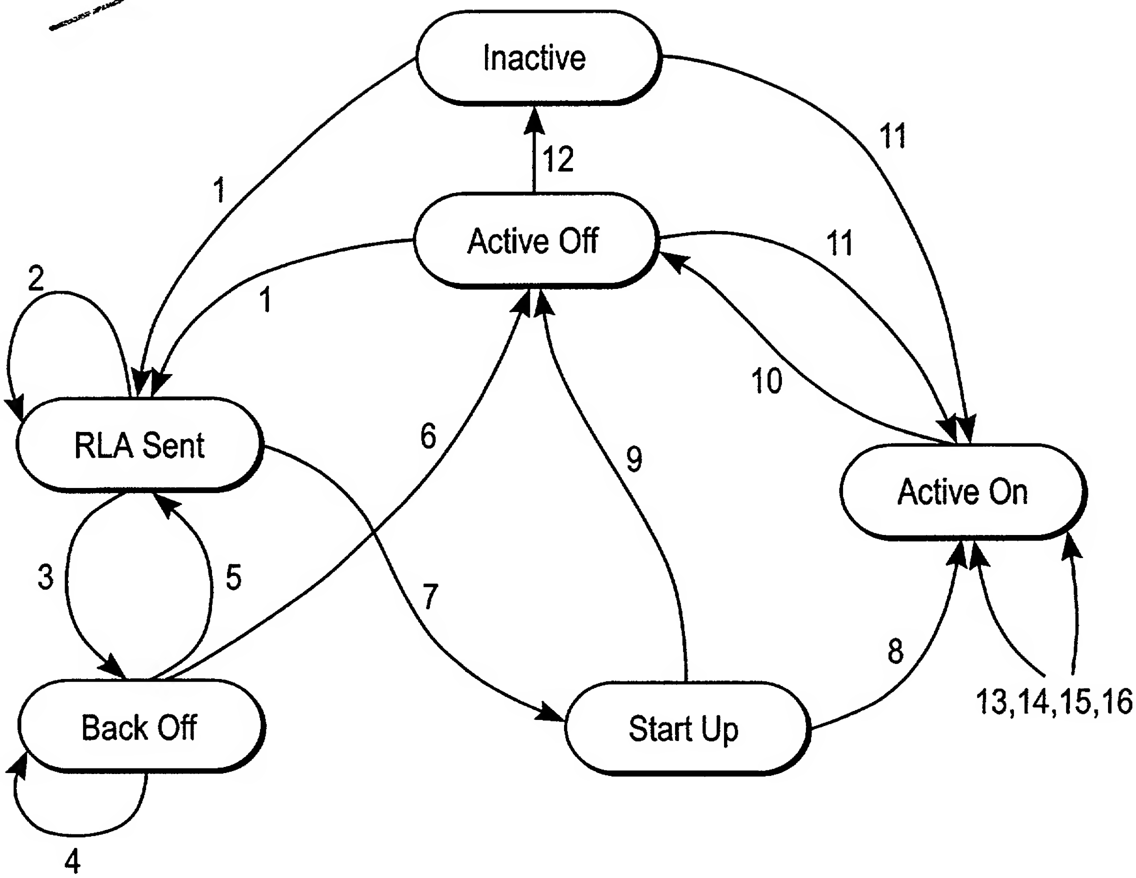

Inactive
12
1
11
Active Off
11
2
1
10
RLA Sent
6
9
Active On
3
5
7
8
13,14,15,16
Back Off
Start Up
4

FIG. 89

1. Event = MA_UnitData.Request
SendRLA; Set ReplyCounter; RLAMiss=0;

2. Event = ReplyTimer > 0
ReplyCounter--;

3. Event = ReplyCounter = 0
RLAMiss++; BORetry--;
BOCounter=Ran(MIN+2^RLAMiss*Win);

4. Event = BOCounter>0
BOCounter--;

5. Event = BOCounter=0 & BORetry>0
RLAMiss=0; SendRLA; Set ReplyCounter

6. Event = BOCounter=0 & BORetry>0
Issue access failure signal; Reset BORetry;

7. Event = FLIRcvd or FLARcvd
Start PE to add partition; wait for partition open

8. Event = PE Success

9. Event = PE Fail
Issue access failure signal (?)

10. Event = Delete last partition
Start PE to delete partition;

11. Event = FLIRcvd or FLARcvd
Start PE to add partition

12. Event = ActiveOffTimeout
Reinitialize encryption/scrambling engines (call PE)

13.Event = MA_UnitData.Request
PE_SendData

14. Event = FLBRcvd
PE_UnitData.Indication

FIG. 90

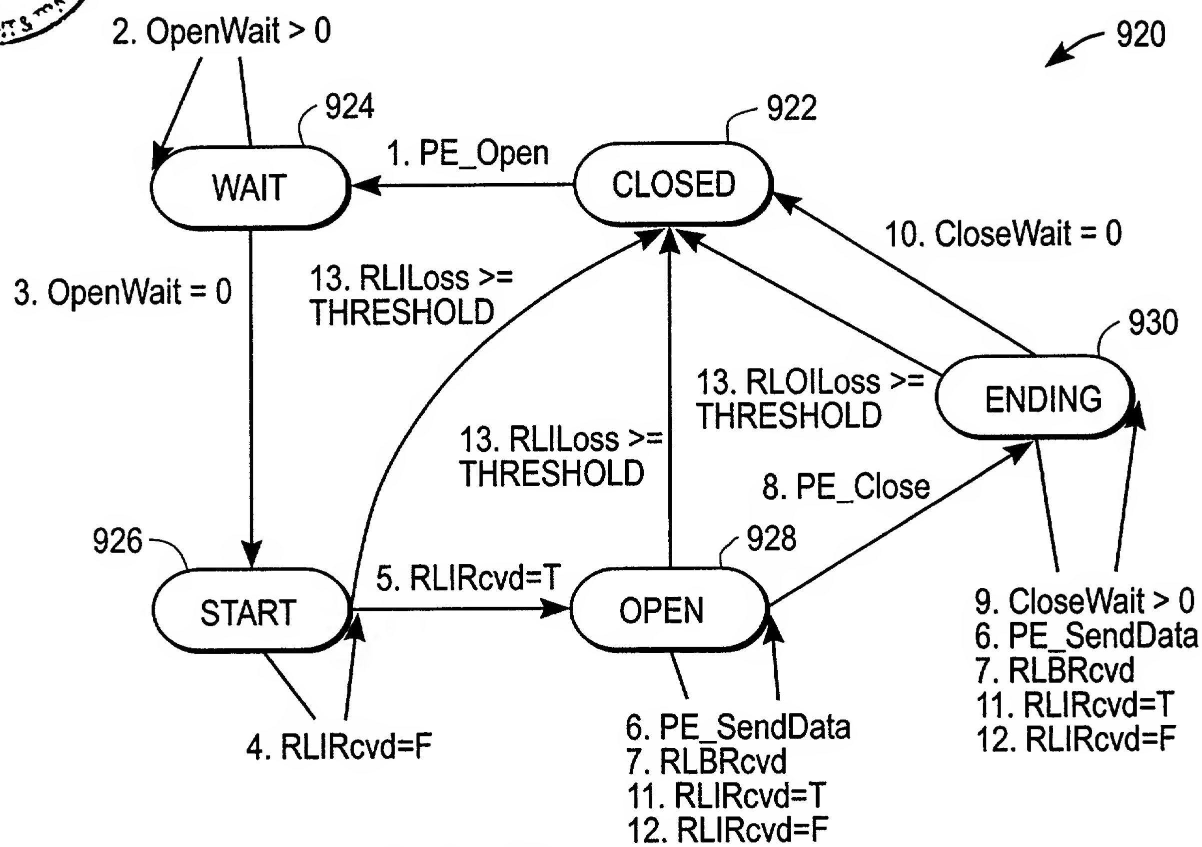

920
2. OpenWait > 0
924
WAIT
1. PE_Open
922
CLOSED
10. CloseWait = 0
3. OpenWait = 0
13. RLILoss >= THRESHOLD
930
ENDING
13. RLOILoss >= THRESHOLD
13. RLILoss >= THRESHOLD
8. PE_Close
926
928
START
5. RLIRcvd=T
OPEN
9. CloseWait > 0
6. PE_SendData
7. RLBRcvd
11. RLIRcvd=T
12. RLIRcvd=F
4. RLIRcvd=F
6. PE_SendData
7. RLBRcvd
11. RLIRcvd=T
12. RLIRcvd=F

FIG. 91

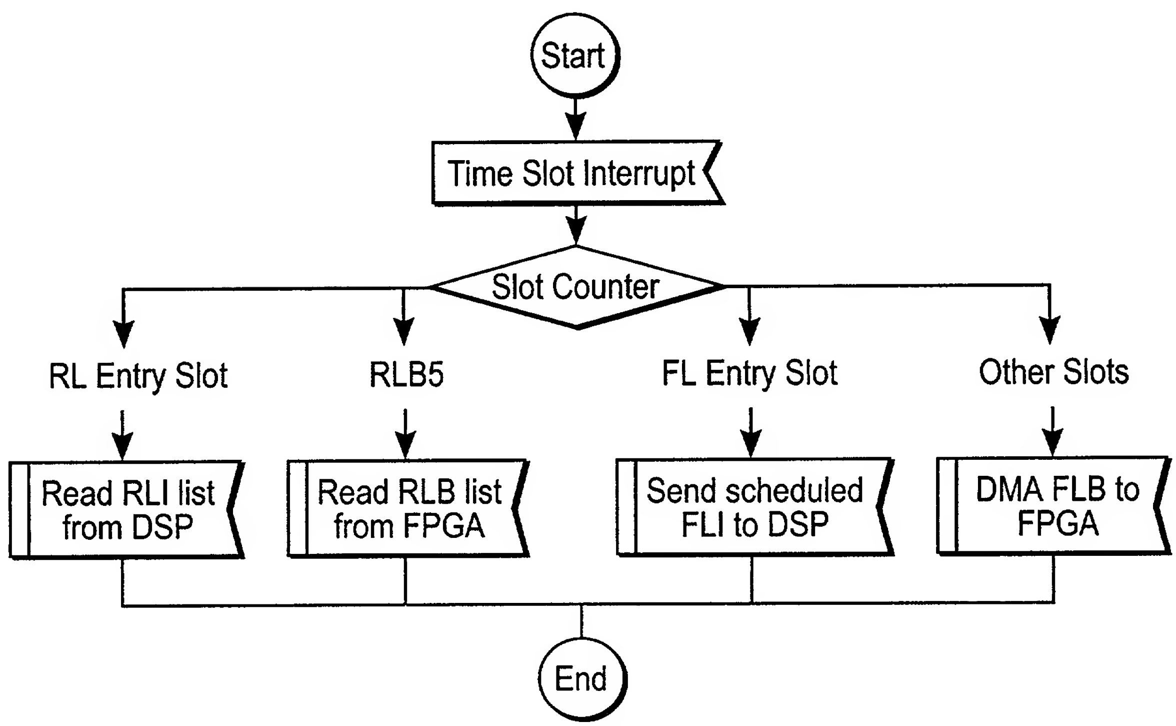

Start
Time Slot Interrupt
Slot Counter
RL Entry Slot
RLB5
FL Entry Slot
Other Slots
Read RLI list from DSP
Read RLB list from FPGA
Send scheduled FLI to DSP
DMA FLB to FPGA
End

FIG. 92

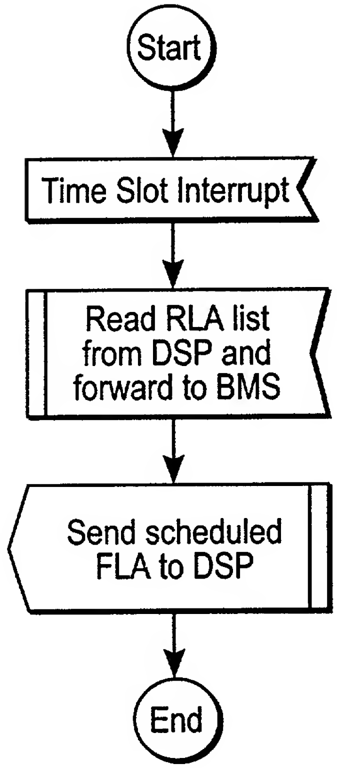


FIG. 93

RUID	RU1	RU5	...	RU500
Backlog	1	12	...	1
Partition 0	1	1		0
Partition 1	0	1		0
...	...	...	...	...
Partition 23	1	0		0
CHANGE	-2	0		+1

FIG. 94

Condition	message	Base starting state	Base action	Prob	Base next state	channel	RU start state	RU action	Prob	RU next state
normal	FLI	closed	send FLI		starting	pass FLI	closed	detect FLI	1-Pm(FLI)	starting
	RLI	starting	detect RLI	1-Pm(RLI)	starting	pass RLI	starting	send RLI		starting
	FLT	starting	send FLT		starting	pass FLT	starting	detect FLT	1-Pm(FLT)	starting
	RLI	starting	detect RLI	1-Pm(RLI)	open	pass RLI	starting	send RLI		starting
	FLT	open	send FLT		open	pass FLT	starting	detect FLT	1-Pm(FLT)	open
			encr = 2 or more	0.99997				encr = 2 or more	0.99458	

Pm_FLI
Pf_FLI
Pm_FLT
Pf_FLT
Pm_RLI
Pf_RLI

FIG. 95

Condition	message	Base starting state	Base action	Prob	Base next state	channel	RU start state	RU action	Prob	RU next state
RU false	FLI	closed	skip FLI		closed	empty	closed	detect FLI	Pf(FLI)	starting
detects FLI	RLI	closed	miss RLI	1	closed	pass RLI	starting	send RLI		starting
and misses	FLT	closed	skip FLT		closed	empty	starting	skips FLT	1-Pf(FLT)	starting
both FLTs	RLI	closed	miss RLI	1	closed	pass RLI	starting	send RLI		starting
	FLT	open	skip FLT		closed	empty	starting	skips FLT	1-Pf(FLT)	closed
			encr = 0	1.00000				encr = 2	9.98E-07	

FIG. 96

Condition	message	Base starting state	Base action	Prob	Base next state	channel	RU start state	RU action	Prob	RU next state
RU misses	FLI	closed	send FLI		starting	pass FLI	closed	detect FLI	1-Pm(FLI)	starting
both FLTs	RLI	starting	detect RLI	1-Pm(RLI)	starting	pass RLI	starting	send RLI		starting
	FLT	starting	send FLT		starting	stop FLT	starting	miss FLT	Pm(FLT)	starting
	RLI	starting	detect RLI	1-Pm(RLI)	open	pass RLI	starting	send RLI		starting
	FLT	open	send FLT		open	stop FLT	starting	miss FLT	Pm(FLT)	closed
			encr = 2	0.99997				encr = 2	4.0E-06	
base misses	FLI	closed	send FLI		starting	pass FLI	closed	detect FLI	1-Pm(FLI)	starting
both RLIs	RLI	starting	miss RLI	Pm(RLI)	starting	stop RLI	starting	send RLI		starting
and RU	FLT	starting	skip FLT		starting	empty	starting	detect FLT	Pf(FLT)	starting
false detects	RLI	starting	miss RLI	Pm(RLI)	closed	stop RLI	starting	send RLI		starting
either FLT	FLT	open	skip FLT		closed	empty	starting	detect FLT	Pf(FLT)	open
			encr = 2	2.9E-05				encr = 2	1.6E-05	
RU misses	FLI	closed	send FLI		starting	stop FLI	closed	miss FLI	Pm(FLI)	closed
FLI	RLI	starting	miss RLI	1-Pf(RLI)	starting	empty	closed	skip RLI		closed
base misses	RLI	starting	miss RLI	1-Pf(RLI)	closed	empty	closed	skip RLI		closed
both RLIs			encr = 2	3.0E-06				encr = 0	3.0E-06	
RU misses	FLI	closed	send FLI		starting	stop FLI	closed	miss FLI	Pm(FLI)	closed
FLI, and base	RLI	starting	detect RLI	Pf(RLI)	starting	empty	closed	skip RLI		closed
false detects	RLI	starting	detect RLI	Pf(RLI)	open	empty	closed	skip RLI		closed
either RLI			encr = 2	8.4E-09				encr = 0	3.0E-06	
RU false	FLI	closed	skip FLI		closed	empty	closed	detect FLI	Pf(FLI)	starting
detects FLI	RLI	closed	miss RLI	1	closed	pass RLI	starting	send RLI		starting
and	FLT	closed	skip FLT		closed	empty	starting	detect FLT	Pf(FLT)	starting
false detects	RLI	closed	miss RLI	1	closed	pass RLI	starting	send RLI		starting
either FLT	FLT	closed	skip FLT		closed	empty	starting	detect FLT	Pf(FLT)	open
			encr = 0	1.00000				encr = 2	9.0E-12	

FIG. 96 (Continued)